(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) EP 1 303 079 A2

(12) EUROPEAN PATENT APPLICATION

(43) Date of publication:
16.04.2003 Bulletin 2003/16

(51) Int Cl.^{7}: H04L 12/24

(21) Application number: 02021812.9

(22) Date of filing: 27.09.2002

(84) Designated Contracting States:
AT BE BG CH CY CZ DE DK EE ES FI FR GB GR
IE IT LI LU MC NL PT SE SK TR
Designated Extension States:
AL LT LV MK RO SI

(30) Priority: 10.10.2001 US 328159 P

(71) Applicant: ALCATEL
75008 Paris (FR)

(72) Inventors:
- Morgan, David
  Salt Lake City, Utah 84124 (US)
- See, Michael
  Chapel Hill, NC 27514 (US)
- Clawson, Steve
  Salt Lake City, Utah 84103 (US)

(74) Representative:
Dreiss, Fuhlendorf, Steimle & Becker
Patentanwälte,
Postfach 10 37 62
70032 Stuttgart (DE)

(54) Central policy based traffic management

(57) A switching node configuring different traffic management protocols via a single centralized set of policies. The switching node includes a central policy repository, a central policy engine, and a central management engine. The central policy repository stores a single set of policies used to manage a plurality of different traffic management protocols in a consistent and predictable manner. The different traffic management protocols may include QoS, NAT, ACL, and the like. The central policy engine evaluates inbound traffic flows based on the policies in the central policy repository, and configures one or more traffic management protocol entities based on a selected policy. The management engine configures and manages the single set of policies via a common set of commands helping to eliminate the danger of creating conflicting policies that may lead to unpredictable results.

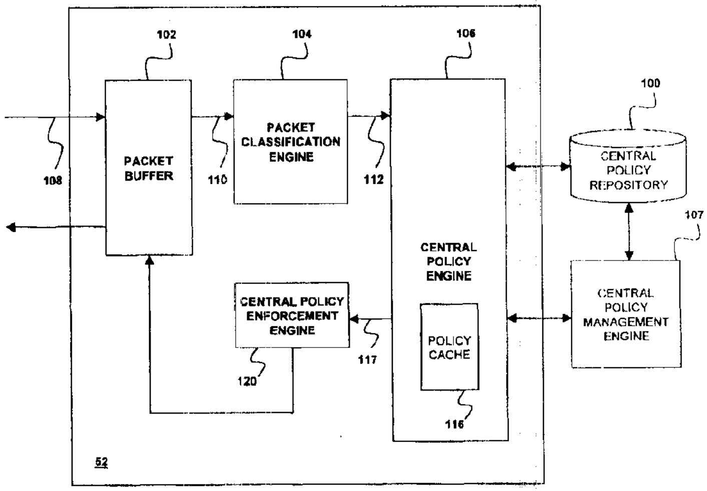

102
104
106
100
PACKET BUFFER
PACKET CLASSIFICATION ENGINE
108
110
112
CENTRAL POLICY REPOSITORY
107
CENTRAL POLICY ENGINE
CENTRAL POLICY ENFORCEMENT ENGINE
117
120
POLICY CACHE
116
CENTRAL POLICY MANAGEMENT ENGINE
52

FIG. 5

Fnnted by Jouve, 75001 PARIS (FR)

Description

CROSS-REFERENCE TO RELATED APPLICATION (S)

[0001] This application claims the benefit of U.S. provisional application No. 60/328,159, filed on October 10, 2001, the content of which is incorporated herein by reference.

FIELD OF THE INVENTION

[0002] The present invention relates generally to traffic control in a data communications network, and more particularly, to configuring different network protocols associated with traffic management via a single centralized set of policies.

BACKGROUND OF THE INVENTION

[0003] Policy based traffic control has become increasingly important in data communication networks as data traffic competes for limited bandwidth. Policy based traffic control generally involves separate, independent interfaces for configuring different traffic management protocols.

[0004] FIG. 1 is a schematic block diagram of a switching node 9 for policy based traffic control according to conventional mechanisms. The switching node includes separate, independent policy engines 11, 13, 15 and associated interfaces 17, 19, 21 for managing and configuring different traffic management protocols. For example, a switching node may include a separate policy engine for controlling quality of service (QoS), IP filtering via access control lists (ACL), network address translation (NAT), and the like. Each policy engine is associated with a control interface used by a network administrator to configure and manage the associated policies for a discrete traffic management protocol.

[0005] In general terms, an inbound packet is processed by a first policy engine, such as, a QoS policy engine, for a matching policy. If a match is found, the matching policy is enforced and the packet is generally not processed by the other policy engines. If a match is not found, the packet is processed by a next policy engine for a match. Under existing technology, therefore, multiple actions from different policy engines are either impossible or difficult to perform for a specific traffic flow. It is often desirable, however, to be able to perform such multiple actions. For example, it may be desirable to invoke a NAT policy engine for address translation based on a source IP address while simultaneously invoking the QoS policy engine for assigning a QoS priority to the flow based on the same source address.

[0006] Another problem with the current policy based traffic control is that the use of separate, independent interfaces for configuring the policies generally requires the network administrator to learn and use different command sequences for configuring and managing different policy engines. This may result in the configuration of conflicting policy rules that may lead to unpredictable results, especially if the configurations are done by different administrators or the same administrator at different times.

[0007] Accordingly, there is a need for a mechanism for providing and applying a common and consistent set of policies to traffic flowing through a data communications network. The mechanism should allow a switching node to enforce a singe policy with multiple actions in a consistent manner.

SUMMARY OF THE INVENTION

[0008] The present invention is directed to traffic management via a single centralized set of policies. According to one embodiment, the invention is directed to a switching node in a data communications network that includes an input, a repository, a policy engine, and a management engine. The repository stores a single set of policies for controlling a plurality of different traffic management protocols. The policy engine is coupled to the input and the repository, and is configured to evaluate an inbound packet based on a policy selected from the single set of policies, and configure one or more traffic management protocol entities based on the selected policy. The management engine is coupled to the repository and the policy engine. The management engine configures and manages the single set of policies via a common set of commands.

[0009] According to another embodiment, the invention is directed to a method for policy based traffic management where the method includes storing in a repository a single set of policies for controlling a plurality of different traffic management protocols. The method includes receiving a first packet, retrieving a first policy from the repository where the first policy identifies a first action for configuring a traffic management protocol entity of a first protocol type, and configuring the traffic management protocol entity based on the first action. The method also includes receiving a second packet, retrieving a second policy from the repository where the second policy identifies a second action for configuring a traffic management protocol entity of a second protocol type, and configuring the traffic management protocol entity based on the second action.

[0010] According to a further embodiment, the invention is directed to a method for policy based traffic management where the method includes storing in a repository a single set of policies for controlling a plurality of different traffic management protocols, receiving a packet, and retrieving a policy from the repository which identifies a first and second action for controlling traffic management protocol entities of a first and second protocol type. The method includes configuring the traffic management protocol entity of the first protocol type based on the first action and configuring the traffic management protocol of the second protocol type based on the second action.

[0011] In an additional embodiment, the invention is directed to a method for policy based traffic management where the method includes storing in a repository a single set of policies for controlling a plurality of different traffic management protocols, receiving a packet, and searching a policy cache for a policy applicable to the packet. If the policy cache does not include an applicable policy, the method includes searching the repository for a match, and generating and storing a new policy in the policy cache. The new policy is selected as applicable to a future packet if the repository contains no other policies that are also applicable to a future packet but are different from the new policy.

[0012] It should be appreciated, therefore, that the evaluation of traffic via a central policy engine allows the configuration of traffic management protocol entities of different protocol types, such as quality of service, access control, network address translation, and the like, in a consistent and predictable manner. The management of the policies via a common set of commands and the storage of the policies in a central repository help eliminate the creation and application of conflicting policies that could result in unpredictable results.

BRIEF DESCRIPTION OF THE DRAWINGS

[0013] These and other features, aspects and advantages of the present invention will be more fully understood when considered with respect to the following detailed description, appended claims, and accompanying drawings where:

FIG. 1 is a schematic block diagram of a switching node for policy based traffic management according to conventional mechanisms;

FIG. 2 is a schematic block diagram of a network environment including a packet switching node according to one embodiment of the invention;

FIG. 3 is a block diagram of a switching interface in one embodiment of the present invention;

FIG. 4 is a schematic block diagram of the packet switching controller providing and applying a common, centralized set of simple policies for controlling a plurality of traffic management protocols according to one embodiment of the invention;

FIG. 5 is a more detailed diagram of the switching controller of FIG. 4 according to one embodiment of the invention;

FIG. 6 is a conceptual layout diagram of various types of policy objects provided by a central management engine for creating a centralized set of policies according to one embodiment of the invention;

FIG. 7 is a conceptual layout diagram of a central policy repository according to one embodiment of the invention; and

FIG. 8 is a flow diagram of a central policy based traffic management according to one embodiment of the invention.

DETAILED DESCRIPTION

[0014] FIG. 2 is a schematic block diagram of a network environment including a packet switching node 10 according to one embodiment of the invention. The packet switching node may also be referred to as a switch, a data communication node or a data communication switch. The packet switching node 10 includes switching interfaces 14, 16 and 18 interconnected to respective groups of LANs 30, 32, 34, and interconnected to one another over data paths 20, 22, 24 via switching backplane 12. The switching backplane 12 preferably includes switching fabric. The switching interfaces may also be coupled to one another over control paths 26 and 28.

[0015] The switching interfaces 14, 16, 18 preferably forward packets to and from their respective groups of LANs 30, 32, 34 in accordance with one or more operative communication protocols, such as, for example, media access control (MAC) bridging and Internet Protocol (IP) routing. The switching node 10 is shown for illustrative purposes only. In practice, packet switching nodes may include more or less than three switching interfaces.

[0016] FIG. 3 is a block diagram of a switching interface 50 in one embodiment of the present invention. The switching interface 50 may be similar, for example, to the switching interfaces 14, 16, 18 of FIG. 2. The switching interface 50 includes an access controller 54 coupled between LANs and a packet switching controller 52. The access controller 54, which may, for example, include a media access controller (MAC), preferably receives inbound packets off LANs, performs flow-independent physical and MAC layer operations on the inbound packets and transmits the inbound packets to the packet switching controller 52 for flow-dependent processing. The access controller 54 also receives outbound packets from the packet switching controller 52 and transmits the packets on LANs. The access controller 54 may also perform physical and MAC layer operations on the outbound packets prior to transmitting them on LANs.

[0017] The packet switching controller 52 receives inbound packets, classifies the packets, modifies the packets in accordance with flow information and transmits the modified packets on switching backplane, such as the switching backplane 12 of FIG. 2. The packet switching controller 52 preferably also receives packets modified by other packet switching controllers via the switching backplane and transmits them to the access controller 54 for forwarding on LANs. The packet switching controller 52 may also subject selected ones of the packets to egress processing prior to transmitting them to the access controller 54 for forwarding on LANs.

[0018] FIG. 4 is a schematic block diagram of the packet switching controller 52 providing and applying a common, centralized set of simple policies for coordinating a plurality of traffic management protocols, such as, for example, access control, address translation, server load balancing, quality of service, and the like. The switching controller 52 includes a central policy engine 56 coupled to a central management engine 58. The central policy engine 56 evaluates the traffic flow against the centralized set of policies to perform, from a central location, one or more policy actions typically performed by separate, independent policy engines. The centralized set of policies include, but are not limited to system policies, network policies, access policies, services policies, and the like. The one or more actions include, but are not limited to packet filtering, packet prioritizing, address translation, server load balance group assignment, and assignment of 802.1p. TOS, or DSCP values.

[0019] The central management engine 58 may include software and/or hardware components to enable a network administrator to configure and manage the centralized set of policies. With the central management engine 58, the network administrator may, from a single location and using a common set of commands, create policies that manage different traffic management protocols.

[0020] FIG. 5 is a more detailed diagram of the switching controller 52 according to one embodiment of the invention. The switching controller includes a packet buffer 102, packet classification engine 104, central policy engine 106, central policy enforcement engine 120, central policy repository 100, and central management engine 107. The packet classification engine 104, central policy engine 106, central policy enforcement engine 120, and central policy management engine 107 are logical devices that may be implemented in software, hardware, firmware (e.g. ASIC), or any combination thereof. It is understood, of course, that FIG. 5 illustrates a block diagram of the packet switching controller without obfuscating inventive aspects of the present invention with additional elements and/or components that may be required for creating the controller. These additional elements and/or components, which are not shown in FIG. 5 are well known to those skilled in the art.

[0021] The switching controller 52 receives inbound packets 108. The packets may include, but are not limited to, Ethernet frames, ATM cells, TCP/IP and/or UDP/IP packets, and may also include other Layer 2 (Data Link/MAC Layer), Layer 3 (Network Layer) or Layer 4 (Transport Layer) data units.

[0022] The received packets are stored in the packet buffer 102. The packet buffer 102 provides via output signal 110 the stored packets or portions thereof to the packet classification engine 104 for processing.

[0023] The packet classification engine 104 may include one or more of a data extractor and a data cache. In an alternative embodiment, the data extractor and data cache are provided within the packet buffer 102.

[0024] The data extractor is used to extract one or more fields from the packets, and to store the extracted fields in the data cache as extracted data. The extracted data may include, but is not limited to, some or all of the packet header. For example, the extracted data may include, but are not limited to, one or more of Layer 2 MAC addresses, 802.1P/Q tag status, Layer 2 encapsulation type, Layer 3 protocol type, Layer 3 addresses, ToS (type of service) values, Layer 4 port numbers, portions of the packet body, and/or any other data used for determining a policy.

[0025] The extracted data is transmitted to the central policy engine 106 via an output signal 112. The central policy engine 106 may be similar to the central policy engine 56 of FIG. 4.

[0026] The central policy engine 106 accesses either an internal policy cache 116 or the central policy repository 100 for selecting a policy applicable to the packet. In accessing the central policy repository 100, the central policy engine 106 communicates with the repository using protocols such as, for example, LDAP.

[0027] According to one embodiment of the invention, the policy cache 116 includes sufficient information for applying policies to existing traffic flows without having to process the entire list of policies in the central policy repository 100 for every packet in the traffic flow. The information in the policy cache 116 is specific enough to prevent packets for which a different applicable rule may exist in the central policy repository 100 to be processed by the policy cache 116.

[0028] The central policy repository 100 may be implemented in a local memory and/or an external directory server with Lightweight Directory Access Protocol (LDAP) access. The central policy repository 100 includes a list of policies that are based on the contents of a packet and/or other elements such as, for example, time information, port information, and the like. In general terms, policies are rules composed of one or more conditions that describe a packet and one or more actions defining how the packet is to be processed if the condition is satisfied.

[0029] The central policy engine 106 compares the extracted packet data with either the policies in the policy cache 116 or central policy repository 100. If a match is found between the condition(s) in the policy and the extracted data, the policy engine determines the action (s) to be taken on the packet. The action(s) to be taken on the packet are transmitted to the central policy enforcement engine 120 via an output signal 117.

[0030] The central policy enforcement engine 120 ensures that the packet is processed according to the parameters defined in the action(s). In this regard, the central policy enforcement engine 120 interacts with other hardware and software elements in the switching node 30 for causing a desired processing of the packet. For example, if the policy actions specify that the packet is to be transmitted at a high priority, the policy enforcement engine 120 may direct the packet buffer 102 to place the packet in a high priority queue of an egress port.

[0031] The central management engine 107 of FIG. 5 may be similar to the central management engine 58 of FIG. 4. The engine 107 may be either a dedicated console or part of a network management console. A network administrator accesses the central management engine 107 for configuring and managing the policies in the central policy repository 100 and the central policy engine 106. According to one embodiment, the central management engine 107 provides a graphical user interface that provides a common set of commands and tools for configuring and managing polices that control different network elements such as, for example, QoS, NAT, ACL, and the like. The central management engine 107 also allows a network administrator to test policies before they are applied.

[0032] FIG. 6 is a conceptual layout diagram of various types of policy objects provided by the central management engine 107 for creating the centralized set of policies according to one embodiment of the invention. In the illustrated embodiment, the policy objects include rule 200 objects, condition 202 objects, action 204 objects, service 206 objects, and group 208 objects. Rules 200 are top level objects including conditions 202 and actions 204. According to one embodiment, rules are provided precedence values indicative of an order in which the rules are to be applied.

[0033] Conditions 202 are parameters used to classify traffic, and actions 204 are parameters describing how to treat the classified traffic. A condition 202 may include a service 206 and/or a group 208. A policy service 206 may be used as a shorthand for certain parts of a condition. According to one embodiment, a policy service 206 is defined by a service name, an IP protocol, a source IP port, and/or a destination IP port. For example, a "video" service may be defined as being associated with a "UDP" protocol and destination IP port number "4500." In another example, a "telnet" service may be defined as being associated with a "TCP" protocol and destination IP port number "23."

[0034] A policy group 208 may be defined by a list of IP/MAC addresses, ports, or services. Policy groups allow a network administrator to define conditions for a group of address, ports, or services, instead of creating a separate condition for each address, port, or service. For example, an "engineering" group may be defined as a set of particular IP addresses. A "basic services" group may be defined as a set of particular services such as telnet, FTP, HTTP, and Sendmail.

[0035] According to one embodiment of the invention, the central management engine 107 allows the creation of policies defining multiple actions for managing different traffic management protocols. For example, a policy in the central policy repository 100 may indicate that traffic with a particular source address and a particular destination address is to have the source address translated and receive a high priority. Thus, two different actions, a NAT policy action and a QoS policy action, may be defined via a single policy rule. FIG. 7 is a conceptual layout diagram of the central policy repository 100 according to one embodiment of the invention. In this illustrated embodiment, the repository includes a policy table 300 including a list of simple policy rules 302. Associated with each policy rule are a precedence number 304, condition 306, and action 308. The precedence number 304 indicates an order in which the rules are to be applied. If traffic matches more than one rule, the central policy engine 106 uses the rule with the highest precedence. In the illustrated embodiment, rule 4 is not matched since it is a subset, or more specific, than the higher precedence rule 2. The precedence ordering of the rules helps eliminate any rule conflicts and ensures that the results of evaluating a traffic flow against the policies is predictable and consistent.

[0036] The condition 306 for each rule defines parameters used for classifying inbound packets. These parameters include but are not limited to individual source addresses or source address groups, individual destination addresses or destination groups, and individual IP protocols 306e, individual IP ports 306d, or policy service groups 306c.

[0037] The action 308 for each rule defines one or more operations to be performed on the packet and/or traffic management protocol entities. The action 308 may be a filtering action, such as for example, dropping or admitting a packet. The action 308 may also be a QoS action such as, for example, assigning a priority to the packet. The action 308 may further be server load balancing, source or destination address translation, mapping or marking 802.1P, TOS, or DSCP values, or a combination of any of the discussed actions.

[0038] FIG. 8 is a flow diagram of a central policy based traffic control according to one embodiment of the invention. The process starts, and in step 400, the packet buffer 102 receives an inbound data packet and stores the packet in the buffer. In step 402, the packet classification engine 104 extracts one or more fields from the packets. In step 404, the central policy engine 106 determines whether the policy cache contains entries that match the extracted fields of the packet. If the answer in YES, the central policy enforcement engine 120 ensures that the policy action indicated in the matched entry of the policy cache is enforced.

[0039] If no match exists in the policy cache 116, the central policy engine 106 determines in step 408 whether there is an exact match of the extracted fields with conditions of a rule in the central policy repository 100. If the answer is YES, the central policy engine proceeds to program, in step 414, the policy cache 116 with the condition fields of the matched policy, the fields having the values of the corresponding extracted fields. This allows future data packets with the same extracted fields to match the newly programmed entry in the policy cache 116, avoiding another search of the central policy repository 100. In step 416, the central policy enforcement engine 120 proceeds to take the policy actions indicated in the matched policy rule.

[0040] If there is no exact match of the conditions of a rule in the central policy repository 100, a determination is made in step 410 whether a partial match exists. If the answer is YES, the central policy engine 106 proceeds to program the policy cache 116 in step 418 with the condition fields of the selected policy, the fields having the values of the corresponding extracted fields, and with a default action. In step 420, the central policy enforcement engine 120 proceeds to take the default policy action.

[0041] If the search of the central policy repository 100 does not result in even a partial match of the rules, the central policy engine 106 proceeds to program the policy cache 116, in step 412, with minimal information needed to forward the packet. According to one embodiment of the invention, such minimal information is a source address of the packet, a destination address of the packet, and a default policy action. In another embodiment of the invention, the minimal necessary information is simply the destination address and the default policy action. The default policy action is enforced by the central policy enforcement engine 120 in step 420.

[0042] The programming of the policy cache 116 will be best understood when considering the following example. Assume that the central policy repository 100 includes the rules depicted in FIG. 7.

[0043] A first packet received by the central policy engine 106 includes the following fields extracted by the classification engine 104:

Source IP - 192.200.200.200
Destination IP - 10.5.3.4
Protocol - TCP
Port - 80 (HTTP)

[0044] The central policy engine 106 compares the extracted information with entries in the policy cache 116. Assuming for purposes of this example that this is a first packet processed by the central policy engine 106, no entries are contained in the policy cache 116.

[0045] The central policy engine 106 then proceeds to search the central policy repository 100 for a match. Upon a finding of no match in the central policy repository 100, the central policy engine programs the policy cache with a minimal amount of information needed to process and forward future similar packets. In one example, the central policy engine may program the policy cache with a source IP address, destination IP address, and a default action. The default action in this example is the assignment of a default priority. The entry placed in the policy cache is as follows:

Source IP - 192.200.200.200
Destination IP - 10.5.3.4
Action - 0 (best effort priority)

[0046] The central policy engine 106 next receives a packet that includes the following fields:

Source IP - 192.200.200.200
Destination IP - 192.168.1.1
Protocol - TCP
Port - 80 (HTTP)

[0047] The central policy engine 106 compares the extracted information with entries in the policy cache 116. Upon a no match, the extracted information is compared against the rules in the central policy repository 100. Rule 1 matches the packet's source IP address, destination IP address, and protocol. However, there is no match in the port information, resulting in a partial match with rule 1.

[0048] In determining an entry for the policy cache, the central policy engine 106 ensures that the entry is specific enough to prevent packets for which a different applicable rule may exist in the central policy repository 100 to be processed by the policy cache. In this regard, the central policy engine 106 determines the number of condition fields of the rule that resulted in the partial match. Rule 1 that resulted in the partial match includes four condition fields: source IP address, destination IP address, protocol and port. Thus, the entry placed in the fast path includes four condition fields although only the source IP and the destination IP are needed to forward future packets. The value of the four fields is based on the information extracted from the packet. Accordingly, the entry placed in the policy cache is as follows:

Source IP - 192.200.200.200
Destination IP - 192.168.1.1
Protocol - TCP
Port - 80 (HTTP)
Action - 0 (best effort priority)

[0049] A next packet received by the central policy engine 106 includes the following fields:

Source IP - 192.200.200.200
Destination IP - 192.168.1.1
Protocol - TCP
Port - 21 (FTP)

[0050] The central policy engine 106 compares the extracted information with entries in the policy cache 116 and does not find a match. If, however, the policy cache had been programmed with less than four fields in processing the previous packet, a match would have resulted in the policy cache, causing the current packet to the forwarded without consulting the rules in the central policy repository.

[0051] The central policy engine 106 proceeds to search the central policy repository 100 and finds an exact match with rule 1. In determining the entry to be programmed in the policy cache, the central policy engine determines the number of condition fields in the rule that resulted in the exact match. The four condition fields of the matching rule 1 are then programmed into the policy cache. The value of the four fields is based on the information extracted from the packet. The entry placed in the policy cache is as follows:

Source IP - 192.200.200.200
Destination IP - 192.168.1.1
Protocol - TCP
Port - 21 (FTP)
Action - 7 (high priority)

[0052] The central policy engine 106 next receives a packet that includes the following fields:

Source IP - 192.200.200.200
Destination IP - 192.168.2.2
Protocol - UDP
Port - 300

[0053] The central policy engine 106 compares the extracted information with entries in the policy cache 116 and does not find a match. The central policy engine 106 then proceeds to search the rules in the central policy repository 100 and finds an exact match with Rule 2. The fields in Rule 2 that resulted in the exact match are source IP address and destination IP address. Accordingly, the entry placed in the policy cache is as follows:

Source IP - 192.200.200.200
Destination IP - 192.168.2.2
Action - 7 (high priority)

[0054] Although this invention has been described in certain specific embodiments, those skilled in the art will have no difficulty devising variations which in no way depart from the scope and spirit of the present invention. It is therefore to be understood that this invention may be practiced otherwise than is specifically described. Thus, the present embodiments of the invention should be considered in all respects as illustrative and not restrictive, the scope of the invention to be indicated by the appended claims and their equivalents rather than the foregoing description.

Claims

1. A switching node in a data communications network, the switching node comprising:

   an input receiving an inbound packet;
   a repository storing a single set of policies for controlling a plurality of different traffic management protocols;
   a policy engine coupled to the input and the repository, the policy engine evaluating the pack et based on a policy selected from the single set of policies and configuring one or more traffic management protocol entities based on the selected policy; and
   a management engine coupled to the repository and the policy engine, the management engine configuring and managing the single set of policies via a common set of commands.

2. The switching node of claim 1, wherein the single set of policies includes a policy identifying two or more actions for controlling two or more traffic management protocols.

3. The switching node of claim 1, wherein the single set of policies includes a first policy identifying a first action for controlling a first traffic management protocol and a second policy identifying a second action for controlling a second traffic management protocol.

4. The switching node of claim 1, wherein one of the traffic management protocols is quality of service.

5. The switching node of claim 1, wherein one of the traffic management protocols is access control.

6. The switching node of claim 1, wherein one of the traffic management protocols is address translation.

7. The switching node of claim 1 further comprising a policy cache coupled to the repository and the policy engine for storing a plurality of cached policies.

8. The switching node of claim 7, wherein the policy engine is configured to evaluate the packet based on the plurality of cached policies prior to evaluating the packet based on the policies in the repository.

9. The switching node of claim 8, wherein the central policy engine selects a cached policy as applicable to the packet if no other policies are stored in the repository that are also applicable to the packet but are different from the selected cached policy.

10. The switching node of claim 8, wherein if no match of a policy is found in the repository, the policy engine is configured to store in the policy cache a policy having a destination address of the packet and a default action.

11. The switching node of claim 8, wherein if a partial match of a policy is found in the repository, the policy engine is configured to store in the policy cache a policy having condition fields of the policy in the repository where the values of the condition fields

are obtained from the packet, and a default action.

12. The switching node of claim 8, wherein if a complete match of a policy is found in the repository, the policy engine is configured to store in the policy cache a policy having condition fields of the policy in the repository where the values of the condition fields are obtained from the packet, and an action indicated by the policy in the repository.

13. A method for policy based traffic management comprising:

    storing in a repository a single set of policies for controlling a plurality of different traffic management protocols;
    receiving a first packet;
    retrieving a first policy from the repository, the first policy identifying a first action for configuring a traffic management protocol entity of a first protocol type;
    configuring the traffic management protocol of the first protocol type based on the first action;
    receiving a second packet;
    retrieving a second policy from the repository, the second policy identifying a second action for configuring a traffic management protocol entity of a second protocol type; and
    configuring the traffic management protocol entity of the second protocol type based on the second action.

14. A method for policy based traffic management comprising:

    storing in a repository a single set of policies for controlling a plurality of different traffic management protocols;
    receiving a packet;
    retrieving a policy from the repository, the policy identifying a first and second action for controlling a first and second traffic management protocol entity, respectively, of a first and second protocol type, respectively;
    configuring the first traffic management protocol entity based on the first action; and
    configuring the second traffic management protocol entity based on the second action.

15. A method for policy based traffic management comprising:

    storing in a repository a single set of policies for controlling a plurality of different traffic management protocols;
    receiving a packet;
    searching a policy cache for a policy applicable to the packet;
    searching the repository if the policy cache does not include an applicable policy; and
    generating and storing a new policy in the policy cache,

    wherein if the new policy is selected as applicable to a future packet, no policies are stored in the repository that are also applicable to the future packet but are different from the new policy.

16. The method of claim 15, wherein if no match of a policy is found in the repository, the policy generated and stored in the policy cache includes a destination address of the packet and a default action.

17. The method of claim 15, wherein if a partial match of a policy is found in the repository, the policy generated and stored in the policy cache includes condition fields of the policy in the repository where the values of the condition fields are obtained from the packet, and a default action.

18. The method of claim 15, wherein if a complete match of a policy is found in the repository, the policy generated and stored in the policy cache includes condition fields of the policy in the repository where the values of the condition fields are obtained from the packet, and an action indicated by the policy in the repository.

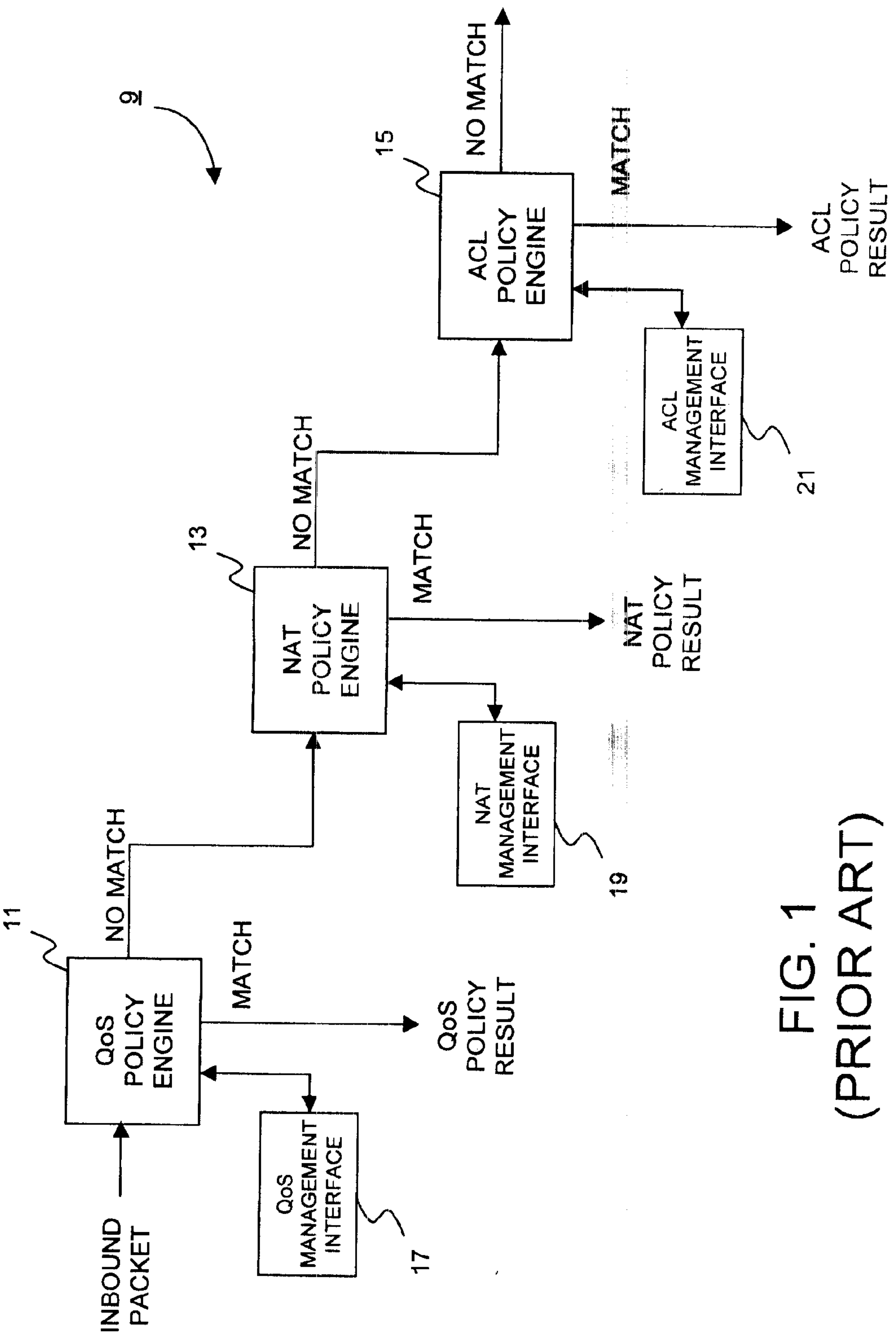


FIG. 1
(PRIOR ART)

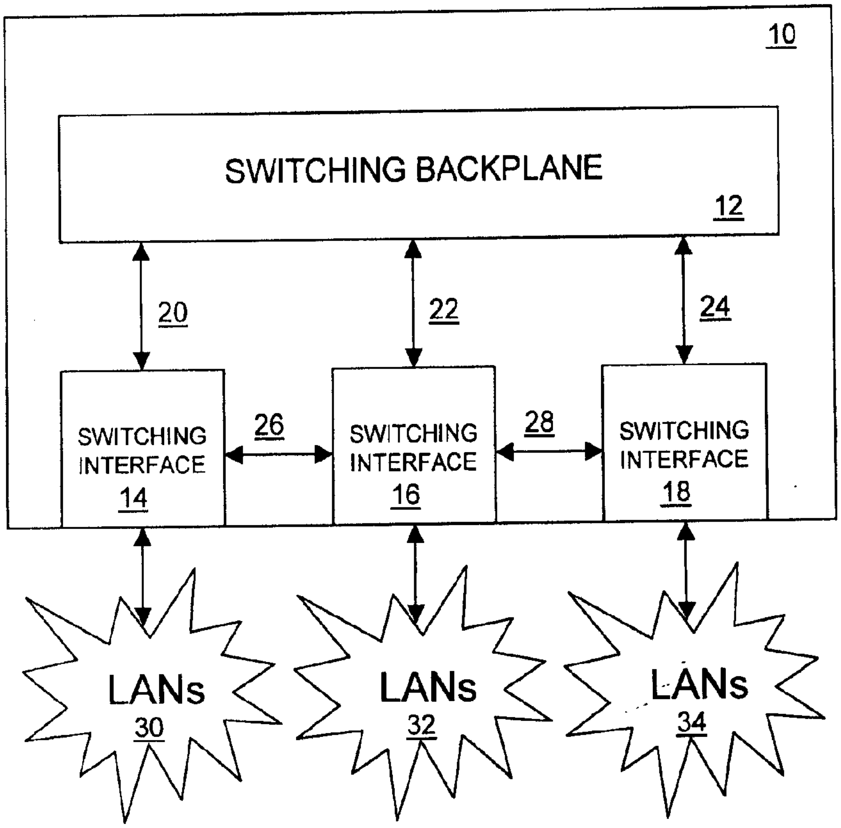

10
SWITCHING BACKPLANE
12
20
22
24
SWITCHING INTERFACE
14
26
SWITCHING INTERFACE
16
28
SWITCHING INTERFACE
18
LANs
30
LANs
32
LANs
34

FIG. 2

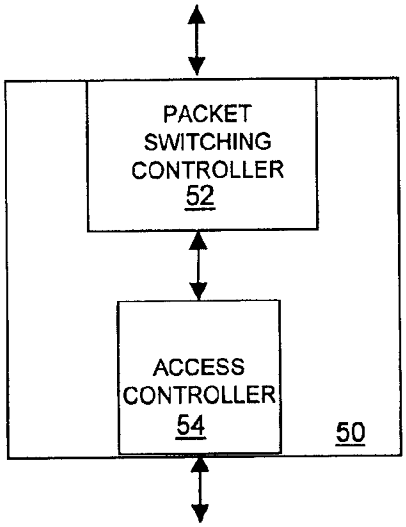

PACKET SWITCHING CONTROLLER
52
ACCESS CONTROLLER
54
50

FIG. 3

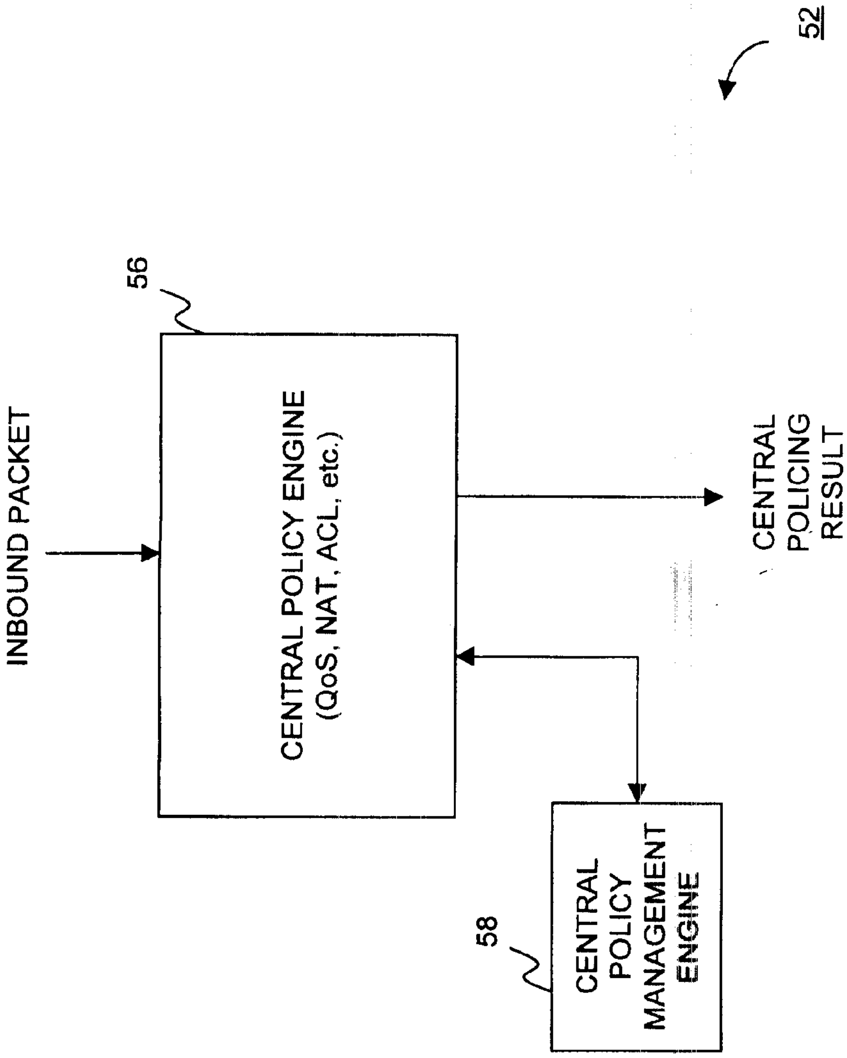

52
INBOUND PACKET
56
CENTRAL POLICY ENGINE
(QoS, NAT, ACL, etc.)
CENTRAL POLICING RESULT
58
CENTRAL POLICY MANAGEMENT ENGINE

FIG. 4

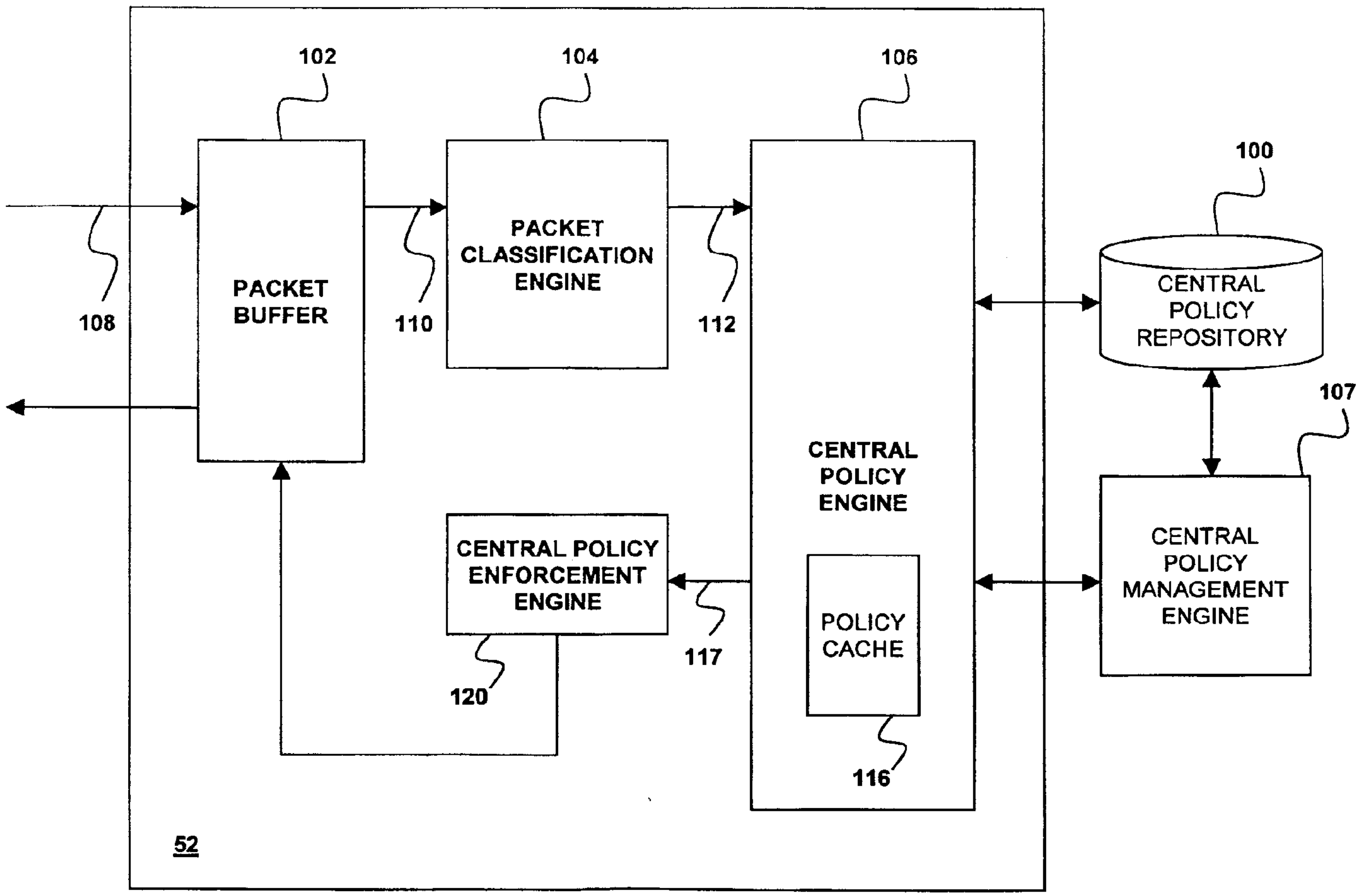

102
104
106
100
PACKET BUFFER
PACKET CLASSIFICATION ENGINE
CENTRAL POLICY ENGINE
CENTRAL POLICY REPOSITORY
108
110
112
107
CENTRAL POLICY ENFORCEMENT ENGINE
POLICY CACHE
CENTRAL POLICY MANAGEMENT ENGINE
117
120
116
52

FIG. 5

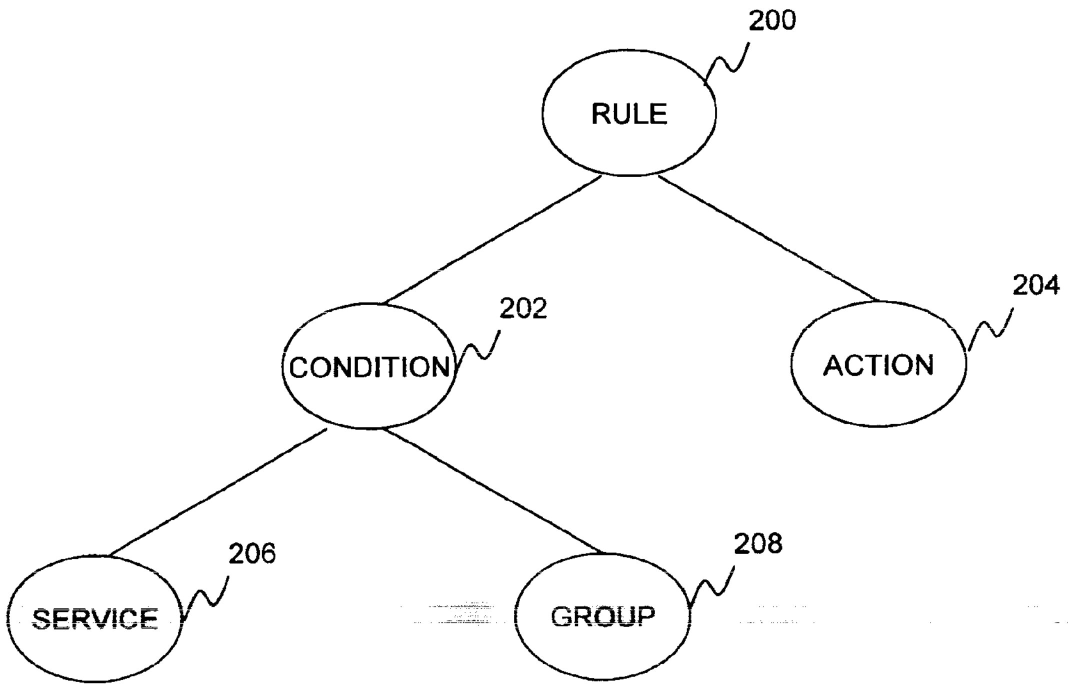
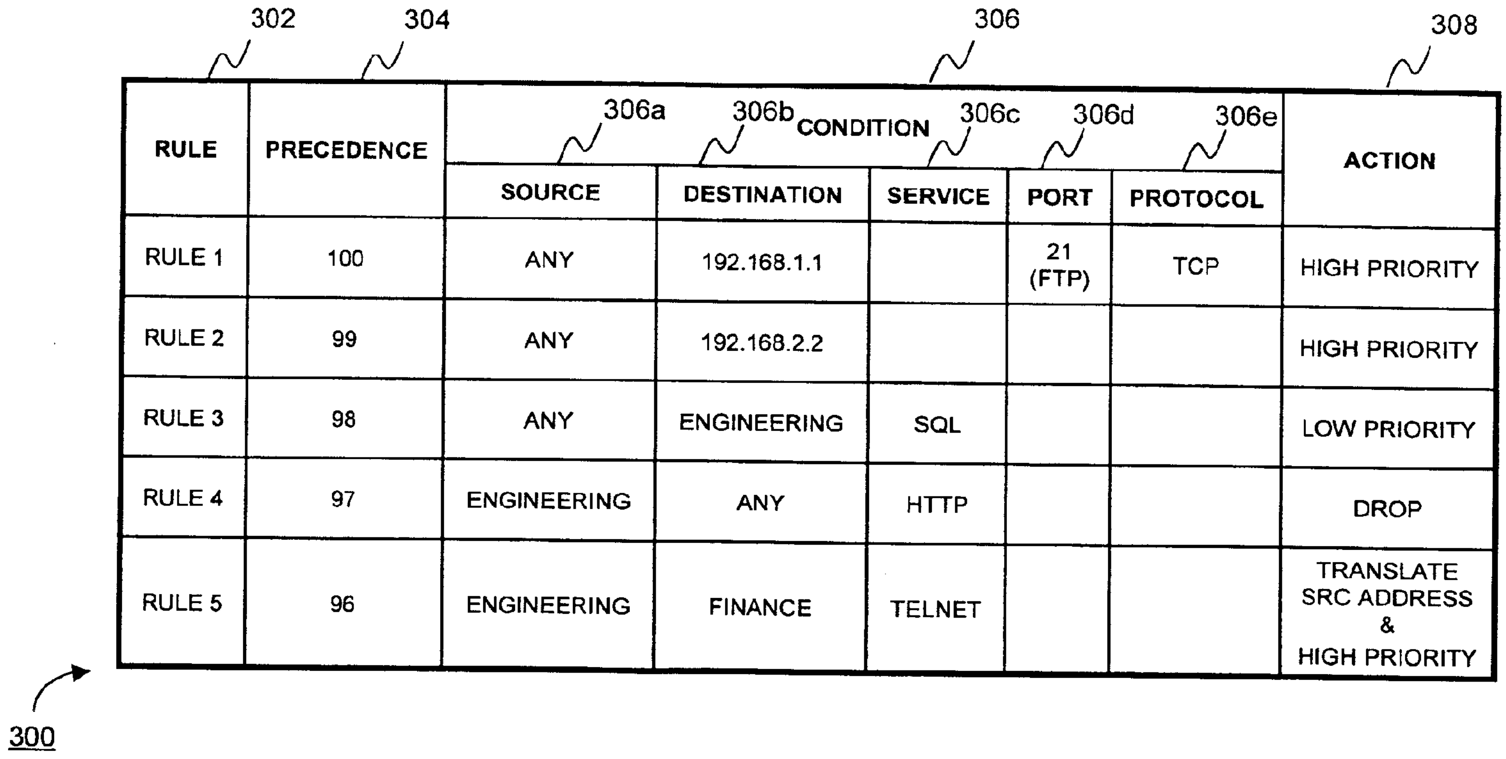


FIG. 6


302
304
306
308
306a
306b
306c
306d
306e
300

RULE	PRECEDENCE	CONDITION					ACTION
		SOURCE	DESTINATION	SERVICE	PORT	PROTOCOL	
RULE 1	100	ANY	192.168.1.1		21 (FTP)	TCP	HIGH PRIORITY
RULE 2	99	ANY	192.168.2.2				HIGH PRIORITY
RULE 3	98	ANY	ENGINEERING	SQL			LOW PRIORITY
RULE 4	97	ENGINEERING	ANY	HTTP			DROP
RULE 5	96	ENGINEERING	FINANCE	TELNET			TRANSLATE SRC ADDRESS & HIGH PRIORITY

FIG. 7

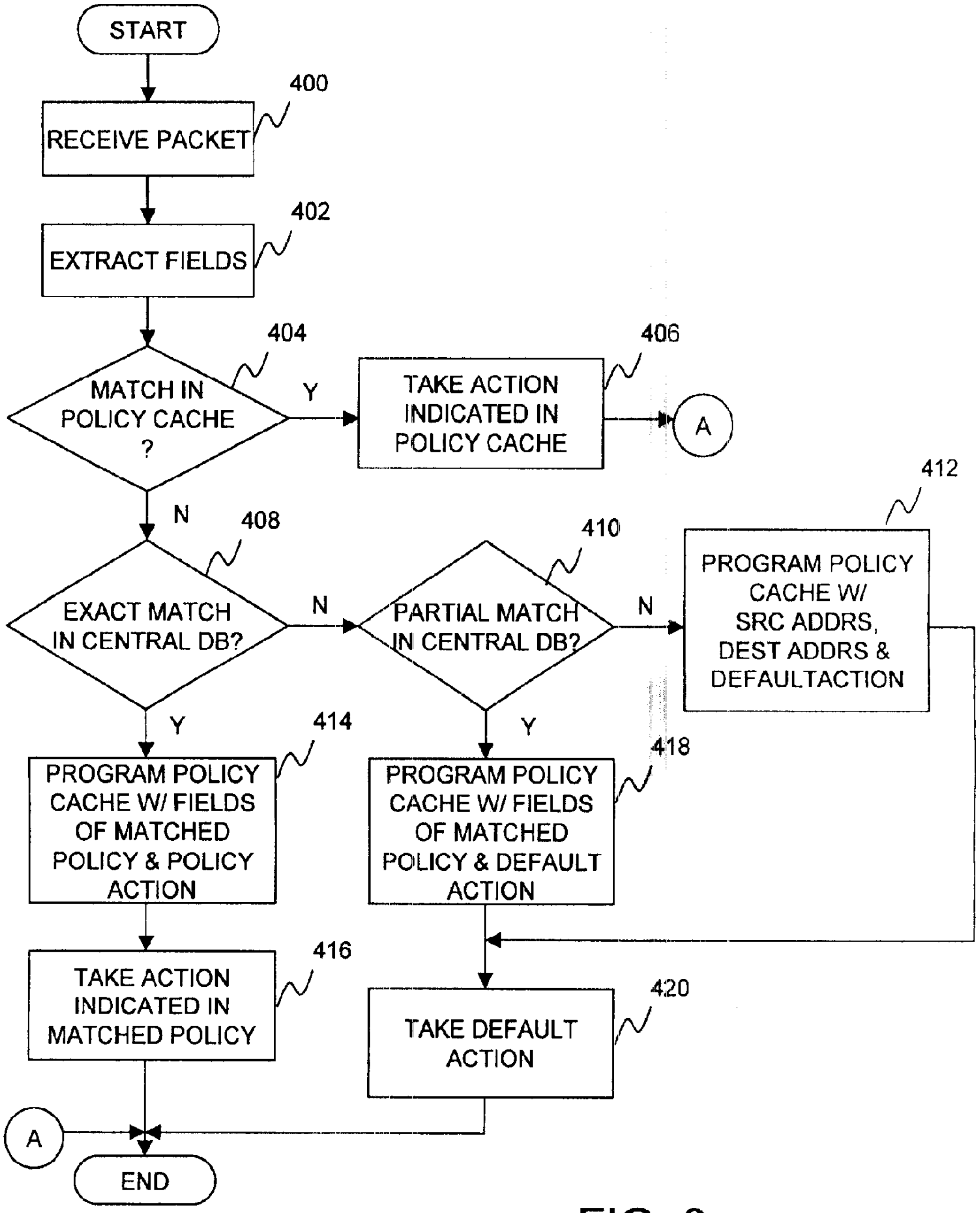

START
RECEIVE PACKET
400
EXTRACT FIELDS
402
MATCH IN POLICY CACHE ?
404
Y
TAKE ACTION INDICATED IN POLICY CACHE
406
A
N
EXACT MATCH IN CENTRAL DB?
408
N
PARTIAL MATCH IN CENTRAL DB?
410
N
PROGRAM POLICY CACHE W/ SRC ADDRS, DEST ADDRS & DEFAULTACTION
412
Y
PROGRAM POLICY CACHE W/ FIELDS OF MATCHED POLICY & POLICY ACTION
414
Y
PROGRAM POLICY CACHE W/ FIELDS OF MATCHED POLICY & DEFAULT ACTION
418
TAKE ACTION INDICATED IN MATCHED POLICY
416
TAKE DEFAULT ACTION
420
A
END

FIG. 8

[19]中华人民共和国国家知识产权局

[51] Int. Cl⁷
H04Q 3/64
H04Q 3/52 H04B 10/12
H04B 10/24 H04B 10/08

[12] 发明专利申请公开说明书

[21] 申请号 01143804.5

[43]公开日 2002 年 7 月 17 日

[11]公开号 CN 1359241A

[22]申请日 2001.12.13 [21]申请号 01143804.5
[30]优先权
[32]2000.12.14 [33]EP [31]00311184.6
[71]申请人 朗迅科技公司
地址 美国新泽西州
[72]发明人 杰瑞恩·沃伦

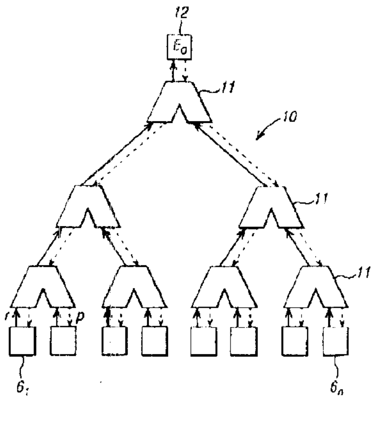
[74]专利代理机构 中国国际贸易促进委员会专利商标事务所
代理人 蒋世迅

权利要求书 3 页 说明书 10 页 附图页数 6 页

[54]发明名称 用于分组交换机和无源光网络的分布式调度器

[57]摘要

一种调度系统和方法，用于从输入端口$(1_1\cdots 1_i)$调度数据分组到输 出端口$(3_1\cdots 3_0)$，它包括虚拟输出队列$(6_1\cdots 6_n)$，虚拟输出队列$(6_1\cdots 6_n)$安排成存储来自输入端口$(1_1\cdots 1_i)$的数据分组，其目的地是一个特 定的输出端口$(3_1\cdots 3_0)$。调度系统包括有多个比较层的调度树(10)，每个比较层安排成成对比较从相关虚拟输出队列$(6_1\cdots 6_n)$并行接收的请 求，以及在剩下单个请求之前，发送较高优先级请求到较高级比较层，单个请求指出被调度的虚拟输出队列$(6_1\cdots 6_n)$发送它的数据分组到相关的输出端口$(3_1\cdots 3_0)$。



知识产权出版社出版

说　　明　　书

用于分组交换机和无源光网络的分布式调度器

技术领域

本发明涉及用于分组交换机的调度器，具体涉及从多个输入端口调度数据分组到至少一个输出端口的方法，该方法包括以下步骤：在多个虚拟输出队列中存储数据分组，虚拟输出队列安排成存储来自多个输入端口中一个输入端口的数据分组，其目的地是该至少一个输出端口中特定的一个输出端口，以及调度多个虚拟输出队列。

背景技术

在太拉比特交换机和吉比特无源光网络（PON）中调度分组要求相当大的计算功率量。若必须部署优先级机构以管理不同服务质量（QoS）的业务，则问题将变得更加复杂。这种复杂性可以表示成系统中每个输出端口需要调度的输入队列总数，即，输入端口数目与服务等级数目的乘积。需要有这样一种算法，能够按照它们具体的优先级调度大量队列中的分组。在最新技术中，即，ASIC 或 FPGA，必须有效地执行这种算法。

C.Blondia，O.Casals 和 J.Garcia 的文章：‘A Cell Based MAC Protocol with Traffic Shaping and a Global FIFO Strategy’，Proceedings of the RACE Open Workshop on Broadband Access，Nijmegen，The Netherlands，June 1993，公开一种媒体接入协议，它利用部署普通的先进先出（FIFO）缓冲器的请求/准许机构。每个网络终端（NT）通过请求广告它的带宽要求，包括 NT 中队列状态信息。利用带宽分配算法，媒体接入协议分配可用的带宽给各个 NT。借助于准许通知 NT 有关分配的带宽。这种用于 PON 的算法（具体地说，异步转移方式（ATM）PON）只寻址少量的队列（~64 个队列），不适合于吉比特容量的大系统（~1000 个队列）。此外，还要求连接 PON 到核心网的附加交换功能。

I.Elhanany，J Nir，D.Sadot 的文章：‘A Contention-Free Packet

Scheduling Scheme for Provision of Quality-of-Service in Tbit/sec WDM Networks’，Optical Networks Magazine, July 2000, 公开一种分组交换机的调度方案。提出的算法声称每个分组时隙周期约 N*^{2}log（N）次操作，其中 N 是输出端口或目的地数目（该文章涉及 N x N 交换机）. 利用循环过程，包括每个输入端口的优先匹配方案以符合各种服务质量的要求，这种方法采用顺序断言不同的输入端口. 对于大量的队列，这种方法仍然太慢。它还不能寻址 PON.

发明内容

本发明试图提供一种用于分组交换机和 PON 的调度器，它能够按照它们特定优先等级调度大量队列中的数据分组。队列的数目等于输入端口的数目，或在管理有不同服务质量要求的数据业务情况下，队列的数目等于输入端口数目与服务等级（或优先等级）数目的乘积。

本发明提供一种按照上述前序部分的方法，其中调度多个虚拟输出队列的步骤包括：借助于调度树，调度与至少一个输出端口中一个端口相关的虚拟输出队列，从而并行地调度与该至少一个输出端口中一个端口相关的虚拟输出队列，调度树至少包括一个比较层，用于执行成对比较从相关虚拟输出队列并行接收的请求，以及在剩下单个请求之前，发送较高优先级请求到较高级比较层，单个请求指出被调度的虚拟输出队列发送它的数据分组到相关的输出端口。

按照本发明的方法有以下的优点，可以有效地调度非常大量的虚拟输出队列。本发明的调度方法只需要 ^{2} log N 次操作，其中 N 是虚拟输出队列的数目。通过级联式接入共享媒体和接入输出端口，该方法不但可以有效地用于分组交换机，还可以用于无源光网络。在相关的分组交换机或无源光网络中所有输出端口，可以并行地执行本发明方法。

在本发明方法的一个实施例中，请求包括识别相关的虚拟输出队列。它允许直接识别准许接入到某个输出端口的虚拟输出队列。

在另一个实施例中，比较层还执行存储较高优先级请求的步骤，在从较高级接收到包括单个请求的准许之后，按照与较高优先级相关的存储请求，发送该准许到较低级比较层。这个实施例可以简化分配机构，避

[19]中华人民共和国专利局

[11] 公开号 CN 1099923A

[12] 发明专利申请公开说明书

[21]申请号 94102225.0

[51]Int.Cl^5
H04Q 3／64

[43]公开日 1995 年 3 月 8 日

[22]申请日 94.3.3

[30]优先权

[32]93.3.3 [33]US[31]08／025,538

[71]申请人 罗姆公司

地址 美国加利福尼亚州

[72]发明人 马克·卡明斯基 罗伯特·佩雷尔曼
彼平·佩特尔 珍妮·伊卡诺斯基
克里斯·袁

[74]专利代理机构 柳沈知识产权律师事务所
代理人 吴秉芬

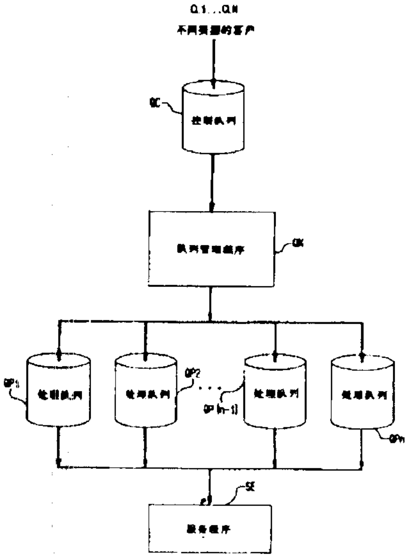
H04M 3／42

说明书页数: 附图页数:

[54]发明名称 队列管理系统和方法

[57]摘要

对多个具有不同客户类型的客户服务的队列管理系统，控制队列以预置顺序对客户进行排列。队列管理程序分配和再分配少于客户类型数的多个处理队列，以与各种客户类型相适应。如果有一相适应的处理队列时该队列管理程序连续地将客户安置该处理队列中，如果不存在相适应的处理队列但有一空的处理队列时该队列管理程序分配或再分配该空的或被空出的处理队列给该客户类型。上述方法可用于语音邮政电话系统的环境中。



为下一个被传送的客户类型。

由于该队列管理程序总是将处理队列已经包含的同一客户类型增补到每个处理队列，该队列管理程序将每个处理队列分配给该队列管理程序首先安置的该处理队列的客户类型。这种分配连续进行直到该服务程序使该处理队列空出为止。如果该队列管理程序现在在该空出的处理队列中安置一新的客户类型，那么该队列管理程序将解除处理队列对前面的客户类型的分配并将它再分配给最新的类型。解除分配和再分配容许一有限的处理队列数目（例如50）去处理任何数量的客户类型（例如1000或更多）。

根据本发明的一特殊方面，即在一电话环境中的条件下，所谓客户是为传送到多个终端而编码的记录语音邮政信息。对于所有终端的所有信息以一预置的顺序出现在该控制队列中，例如以一先入先出为基础。在比其终端数目要少的多个处理队列之间，队列管理程序通过将控制队列中的每个相继的第一信息安置到其终端与该第一信息的终端相适应的处理队列中的方法，一个接一个地分配该控制队列中的信息。如果不存在有相适应的处理队列，但存在一空出的处理队列时，则该队列管理程序将该第一信息置入该空出的处理队列。如果不存在空出的处理队列，则该队列管理程序在控制队列中保持该信息直至所有的处理队列的信息传送而使一处理队列空出为止。该队列管理程序这时将该处理队列再分配仅用来传送到新的终端的信息并将下一个信息安置到该空出的处理队列。

本发明使用了有限数量的队列来提供同时对大量的客户或终端的服务。实际上对不同客户类型或终端并设有限制。用于同一终端的同一类型的客户或信息是在同一队列中等待的。在同一队列中等

待的客户或信息，虽然可能有不同的排序，但在正常状态下是以年月日顺序排列的。对不同客户类型或信息终端的同时处理可容易实现。该系统动态地将队列分配给每个客户类型或终端，并当它们变为空出以及需用新终端的新的客户类型或信息到来时将它们重新分配。为了处理容易起见，本发明将与一给定客户类型或终端相关的处理与相应的队列关联起来。

对于多个客户类型或用于多个终端的信息的处理比用于一单一队列方案及对上述多个队列系统的处理变得更为有效。本发明考虑到在处理中有大的适应性。

本发明的这些和其它的特征在构成本说明书的一部分的权利要求中指出。本发明的其它目的和优点为借助于附图阅读时对本领域的技术人员来说将变得显然。

图1是一说明体现本发明特征的队列管理系统的方框图；

图2是说明在图1中所示的并体现了本发明的队列管理系统的操作流程图；

图3是一说明在图1中所示的并体现了本发明的队列管理系统的操作的另一种方法的流程图；

图4是一说明当它们开始图1所示的处理队列的服务时该服务程序的操作的流程图；

图5是说明当它们进行在该队列中的操作时在图1中所示的服务程序的操作的另外的流程图；

图6是一体现本发明的一实施例的一电话系统的方框图；

图7是在图6所示的电话系统的环境中体现本发明的一队列管理系统的一方框图；

US005920568A

United States Patent [19]

Kurita et al.

[11] **Patent Number: 5,920,568**

[45] **Date of Patent: Jul. 6, 1999**

[54] **SCHEDULING APPARATUS AND SCHEDULING METHOD**

[75] Inventors: **Toshihiko Kurita**; **Ichiro Iida**, both of Kanagawa, Japan

[73] Assignee: **Fujitsu Limited**, Kanagwa, Japan

[21] Appl. No.: **08/790,443**

[22] Filed: **Jan. 29, 1997**

[30] **Foreign Application Priority Data**

Jun. 17, 1996 [JP] Japan 8-155746

[51] **Int. Cl.**^{6} **H04L 12/28**; H04L 12/56

[52] **U.S. Cl.** **370/412**; 370/411; 370/429

[58] **Field of Search** 370/414, 416–418, 370/395, 468, 229, 335, 351, 428, 429, 411, 412

[56] **References Cited**

U.S. PATENT DOCUMENTS

5,231,633	7/1993	Hluchyj et al.	370/94
5,268,900	12/1993	Hluchyj et al.	370/94
5,499,238	3/1996	Shon	370/411

Primary Examiner—Chi H. Pham
Assistant Examiner—Afsar M. Qureshi
Attorney, Agent, or Firm—Helfgott & Karas, P.C.

[57] **ABSTRACT**

A scheduling apparatus and a scheduling method are capable of reading data elements from a plurality of queues in such a form that past hysteresis reflects therein. The scheduling apparatus comprises a queue hysteresis table for storing a value e_count obtained subtracting the number of data elements (packets in a router) actually fetched out of the queue, from the number of times with which this queue becomes a processing target with respect to each queue. The apparatus also comprises a scheduling unit for cyclically designating each queue as a processing target, adding "1" to e_count, corresponding to that queue, in the queue hysteresis table if no data elements exist in the queue designated as the processing target, consecutively fetching, from the processing target queue, the data elements the number of which corresponds to a value of e_count corresponding to the queue if the data elements exist in the processing target queue, and decrementing the value of e_count by the number of fetched data elements.

10 Claims, 13 Drawing Sheets

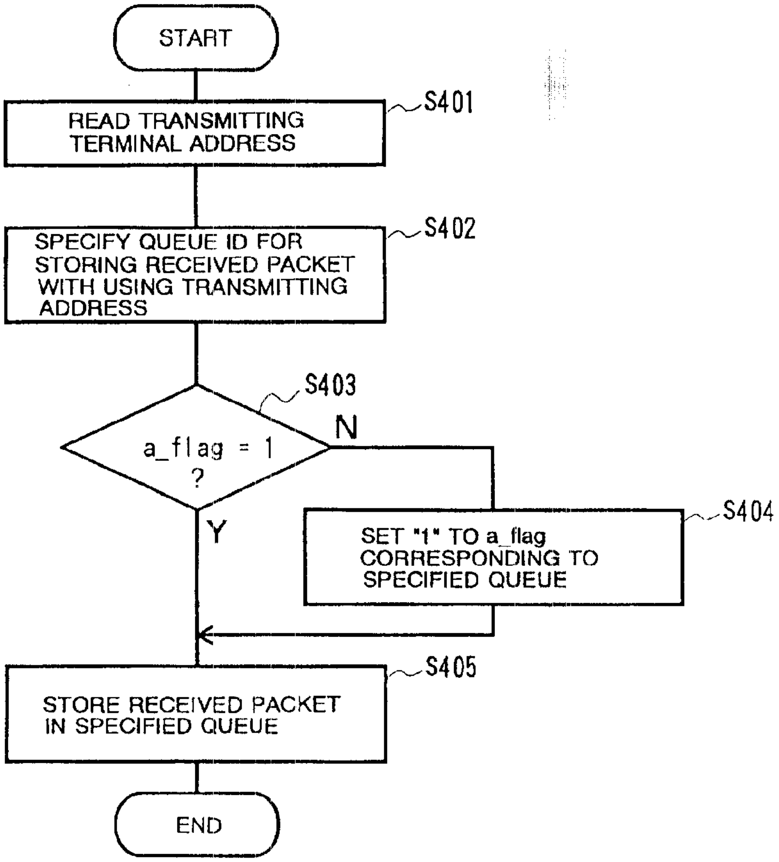

START
READ TRANSMITTING TERMINAL ADDRESS
S401
SPECIFY QUEUE ID FOR STORING RECEIVED PACKET WITH USING TRANSMITTING ADDRESS
S402
S403
a_flag = 1 ?
N
Y
SET "1" TO a_flag CORRESPONDING TO SPECIFIED QUEUE
S404
STORE RECEIVED PACKET IN SPECIFIED QUEUE
S405
END

FIG. 1

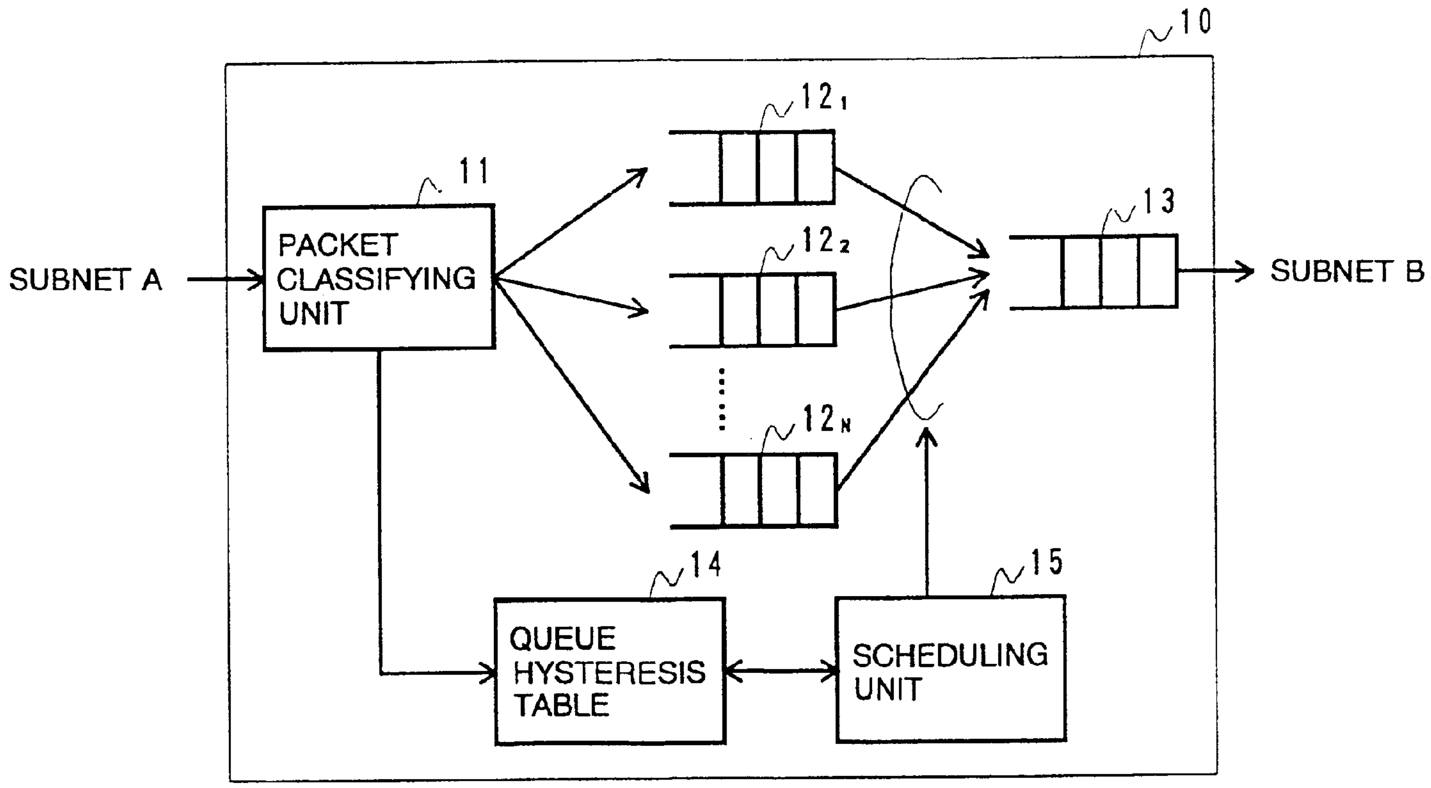

10
SUBNET A
11
PACKET CLASSIFYING UNIT
12₁
12₂
12ɴ
13
SUBNET B
14
QUEUE HYSTERESIS TABLE
15
SCHEDULING UNIT

FIG. **8** is an explanatory diagram showing an outline of the queue hysteresis table incorporated into the router in a second embodiment;

FIG. **9** is a flowchart showing operation procedures of the scheduling unit provided in the router in the second embodiment;

FIG. **10** is an explanatory diagram showing an outline of the queue hysteresis table incorporated into the router in a third embodiment;

FIG. **11** is a flowchart showing operation procedures of the scheduling unit provided in the router in the third embodiment;

FIG. **12** is a diagram illustrating an example of internet working that uses the router;

FIG. **13** is a block diagram showing functions of a router employing a prior art fair queuing method; and

FIG. **14** is an explanatory diagram showing scheduling procedures by the router using the prior art fair queuing method.

DESCRIPTION OF THE PREFERRED EMBODIMENTS

The present invention will hereinafter be specifically described with reference to the accompanying drawings.

First Embodiment

FIG. **1** illustrates an outline of a configuration of a router to which a scheduling method (and a scheduling apparatus) of the present invention is applied. A router **10** in a first embodiment is an apparatus for connecting a subnet A to a subnet B, and includes, as shown in FIG. **1**, a packet classifying unit **11**, queues 12_1–12_N, and output buffer **13**, a queue hysteresis table **14** and a scheduling unit **15**. Note that, the router **10** is constructed so that some queues **12** are created in a memory dynamically in accordance with a receiving condition of packets. Therefore, queues 12_1–12_N are not always exist in the router **10**. Although in this embodiment, assuming, for the sake of convenience, that the router **10** is provided with N queues **12** which functions independently, the configuration and operation will be discussed first. Thereafter, real configuration and operation of the router **10** will be discussed.

The queues 12_1–12_N are buffers for temporarily storing packets that should be transmitted to the subnet B, and are each made corresponding to transmitting stations (terminals) connected to the subnet A. The packet classifying unit **11** supplies the queues **12** corresponding to transmitting addresses contained in those packets with packets received from the subnet A. Note that the router **10** in this embodiment is constructed so as to transfer packets (so-called internet packet) each having a structure shown in FIG. **2**. Hence, the packet classifying unit **11** specifies a queue **12** to store the received packet by reading the transmitting terminal address (transmitting terminal IP address) included in the header of the received packet.

Further, the packet classifying unit **11** executes a process of rewriting contents of the queue hysteresis table **14** (the details of which will be stated later on) in parallel to the above-described supplying process of the packets. The output buffer **13** is temporarily stored with the packets within the respective queues **12**, and the packets in the output buffer **13** are transmitted in the as-inputted sequence to the subnet B.

The queue hysteresis table **14** is a table the contents of which are updated by the packet classifying unit **11** and the scheduling unit **15**. As illustrated in FIG. **3**, the queue hysteresis table **14** is stored with active flags (a_flag) and empty count numbers (e_count) in such a form that these flags and numbers are made corresponding to queue IDs defined as data for identifying the queues.

The packet classifying unit **11** rewrites values of the active flags a_flag within the queue hysteresis table **14** in accordance with a packet receiving condition. More specifically, when receiving a packet, the packet classifying unit **11** reads, as shown in FIG. **4**, a transmitting terminal address included in the header of the received packet first (step **S401**). Next, the packet classifying unit **11** specifies a queue ID for storing the received packet by referring to a address-queue ID table which holds the relationship between the transmitting terminal addresses and the queue IDs, and is provided inside the unit (step **S402**). Thereafter, the packet classifying unit **11** checks the value of the active flag corresponding to the specified queue ID in the queue hysteresis table **14** (step **S403**).

When the value of the active flag is "0" (step **S403**;N), the packet classifying unit **11** rewrites the active flag to "1" (step **S404**). Then, the packet classifying unit **11** carries out a process for storing the received packet in the specified queue **12** (step **S405**). On the contrary, when the active flag is "1" (step **S403**;Y), the packet classifying unit **11** carries out a process for storing the received packet in the specified queue **12** (step **S405**) without rewriting the active flag.

Moreover, the packet classifying unit **11**, parallel to (independently on) the series of the above described processes, performs a process for managing receiving time of the last received packet for each queue ID the active flag of which is set to "1". Then, the packet classifying unit **11** changes, based on the managing results, the active flag concerning the queue ID of which the receiving time becomes before a time which a predetermined time (such as 60 sec.) is subtracted from a current time to "0".

The scheduling unit **15** performs control to transmitting the packets stored in the queues **12** to the output buffer **13**. The scheduling unit **15**, when executing this control, refers to values of the flags a_flag in the queue hysteresis table **14**, and refers to and updates values of the empty count numbers e_count.

Operations of the router and the scheduling unit **15** in this embodiment will hereinafter be described specifically.

FIG. **5** is a flowchart showing operation procedures of the scheduling unit **15** after starting up the router. As shown in FIG. **5**, the scheduling unit **15**, when actuating this router, to begin with, sets "1" in variables i and j respectively, and initializes the contents of the queue hysteresis table **14** (step **S101**). In step **S101**, the scheduling unit **15** sets "0" in all of the flags a_flag and of the empty count numbers e_count, thereby initializing the queue hysteresis table **14**. Moreover, the packet classifying unit **11**, after the scheduling unit **15** has executed step **S101**, starts executing the above-mentioned processes (of supplying the respective queues with the packets and updating the values of a_flag in the queue hysteresis table **14**).

After initializing the queue hysteresis table **14**, the scheduling unit **15** judges whether an active flag a_flag, of the queue the queue ID of which is i, is "1" or not (step **S102**). If a_flagis not "1", (step **S102**; N), the scheduling unit **15** executes a process (steps **S120**–**S122**) for setting the queue ID of the next queue in the variable i. That is, the scheduling unit **15** compares the value of the variable i with a maximum value N of the queue ID and, if the value of the variable i is not coincident with N (step **S120**; N), adds "1" to the variable i (step **S121**). Whereas if i is coincident with N (step **S120**; Y), the scheduling unit **15** sets "1" in the variable i (step **S122**).

到启示，将上述特征应用于对比文件1中，也就是在对比文件1的基础上结合对比文件2，从而得到权利要求3请求保护的技术方案，对于本领域技术人员而言是显而易见的。因此，权利要求3不具有突出的实质性特点，不具备专利法第二十二条第三款规定的创造性。

4、权利要求4引用权利要求1，其附加技术特征“检测何时队列为空；和回收所述空队列”在对比文件1中没有被公开，为权利要求4与对比文件1相比区别技术特征，由此确定实际所要解决的技术问题在于：节约存储队列的缓存空间。对比文件3（CN1099923A）公开了一种队列管理方法（参见说明书第3页12行至第4页6行，摘要）：系统动态地将分配队列，当检测到队列为空时，对其回收重新分配。由此可见，上述区别技术特征已经被对比文件3公开，且该特征在对比文件3中所起地作用与其在本发明中为解决其技术问题所起的作用相同，都是用于节约存储队列的缓存空间。因此，本领域技术人员可以从对比文件3中得到启示，将上述特征应用于对比文件1中，也就是说在对比文件1的基础上结合对比文件3，从而得到权利要求4请求保护的技术方案，对于本领域技术人员而言是显而易见的。因此，权利要求4不具有突出的实质性特点，不具备创造性。

5、权利要求6引用权利要求1。对比文件1中已经披露了（参见说明书第7栏55-58行）分类是基于分组特定源地址、特定目的地址或优先级的。由此可见，权利要求6附加的技术特征也已经被对比文件1公开。因此，当其引用的权利要求1不具有新颖性时，权利要求6也不具备新颖性。

6、权利要求11请求保护一种网络设备。对比文件1（EP1303079A2）已经公开了一种在交换节点的网络设备（参见摘要，说明书第1栏41行至第2栏42行，第7栏19-26行，第7栏56行至第8栏6行，权利要求1、3、4、7，图3-10）：其对入站分组（相当于权利要求11中的进入分组）进行分类（相当于具有权利要求11中用于对进入分组进行分类的装置的功能模块），确定这些分类是否已经分配了策略队列（相当于具有权利要求11中用于确定队列是否已被分配给所述分类的装置的功能模块），在策略队列还没被分配给相应分类时分配这些策略队列（相当于具有权利要求11中用于在所述队列还未被分配给所述分类时分配所述队列的装置）。由此可见，权利要求11的全部内容已经在对比文件1中公开，两者采用相同的技术方案，且均涉及网络设备流量控制

领域，都解决了高效分配分组队列的技术问题，并达到相同的预期效果。因此，权利要求11不具有专利法第二十二条第二款规定的新颖性。

7、权利要求12引用权利要求11。对比文件1中已经披露（参见说明书第4栏31行至第5栏18行，图3-4）网络设备的交换端口50（相当于权利要求12中的入口端口）中的分组交换控制器52与策略队列相关联。由此可见，权利要求12附加的技术特征已经被对比文件1公开。因此，当其引用的权利要求11不具有新颖性时，权利要求12也不具备新颖性。

8、权利要求13引用权利要求11，其附加技术特征“其中所述队列是虚拟输出队列”在对比文件1中没有被公开，为权利要求13与对比文件1相比区别技术特征，由此确定实际所要解决的技术问题在于：不受物理输出队列制约，提高输出队列的数目及有效性。对比文件2（CN1359241A）公开了一种分组交换机（参见说明书第2-3页）：利用虚拟输出队列（相当于本申请权利要求13中的虚拟输出队列）发送数据分组到相关的输出端口。由此可见，上述区别技术特征已经被对比文件2公开，且该特征在对比文件2中所起的作用与其在本发明中为解决其技术问题所起的作用相同，都是用于实现有效地调度非常大量的输出队列。因此，本领域技术人员可以从对比文件2中得到启示，将上述特征应用于对比文件1中，也就是在对比文件1的基础上结合对比文件2，从而得到权利要求13请求保护的技术方案，对于本领域技术人员而言是显而易见的。因此，权利要求13不具有突出的实质性特点，不具备专利法第二十二条第三款规定的创造性。

9、权利要求14引用权利要求11，其附加技术特征“用于检测何时队列为空的装置；和用于回收所述空队列的装置”在对比文件1中没有被公开，为权利要求14与对比文件1相比区别技术特征，由此确定实际所要解决的技术问题在于：节约存储队列的缓存空间。对比文件3（CN1099923A）公开了一种队列管理系统（参见说明书第3页12行至第4页6行，摘要）：系统动态地将分配队列，当检测到队列为空时（相当于具有权利要求14中用于检测何时队列为空的装置的功能模块），对其回收重新分配（相当于具有权利要求14中用于回收所述空队列的装置的功能模块）。由此可见，上述区别技术特征已经被对比文件3公开，且该特征在对比文件3中所起地作用与其在本发明中为解决其技术问题所起的作用相同，都是用于节约存储队列的缓存空间。因此，本领域技

术人员可以从对比文件3中得到启示，将上述特征应用于对比文件1中，也就是说在对比文件1的基础上结合对比文件3，从而得到权利要求14请求保护的技术方案，对于本领域技术人员而言是显而易见的。因此，权利要求14不具有突出的实质性特点，不具备创造性。

10、权利要求16引用权利要求11。对比文件1中已经披露了（参见说明书第7栏55-58行）分类是基于分组特定源地址、特定目的地址或优先级的。由此可见，权利要求16附加的技术特征也已经被对比文件1公开。因此，当其引用的权利要求11不具有新颖性时，权利要求16也不具备新颖性。

11、权利要求26请求保护一种在网络设备中分配队列的方法。对比文件4（US5920568A）公开了一种在路由器中分配队列的方法（参见说明书第5栏第26行至第6栏第41行，图1）：队列并不是一直存在的，而是根据接收到分组动态创建的（即初始时刻没有分配队列），接收一个分组，对其进行分类（相当于权利要求26中接收第一分组，对所述第一分组进行分类），根据该分组状态将一个队列分配给该分组（相当于权利要求26中将第一队列分配给所述第一分类），接收下一个分组，对其进行分类（相当于权利要求26中接收第二分组，对所述第二分组进行第二分类），根据分组状态确定是否与之前分组分类相同（相当于权利要求26中确定所述第一分类是否与所述第二分类相同）。由此可见，权利要求26的全部内容已经在对比文件4中公开，两者采用相同的技术方案，且均涉及网络设备流量控制领域，都解决了对分组分类并分配队列的技术问题，并达到相同的预期效果。因此，权利要求26不具有专利法第二十二条第二款规定的新颖性。

12、权利要求27引用权利要求26。对比文件4中已经披露（参见说明书第6栏8-14行）分组分类单元根据分组状态不同将其分配到不同的队列（相当于权利要求27中在所述第一分组不同于所述第二分组时分配第二队列）。由此可见，权利要求27附加的技术特征也已经被对比文件4公开。因此，当其引用的权利要求26不具有新颖性时，权利要求27也不具备新颖性。

13、权利要求28引用权利要求26，其附加技术特征“还包括在所述第一分类不同于所述第二分类时将所述第二分组分派给所述第一队列的步骤”在对比文件4中没有

被公开，为权利要求28与对比文件4相比区别技术特征。由此确定实际所要解决的技术问题是统一队列发送分组。而在网络设备通常出现仅有一个输出缓冲队列（物理或虚拟的）可用的情形，则本领域技术人员容易想到这时无论待发送的分组属于什么类型都只能分配到该缓冲队列中，统一发送，这是本领域的惯用手段。由此可见，在对比文件4的基础上结合本领域的惯用手段，从而得到权利要求28，对于本领域技术人员而言是显而易见的。因此，权利要求28不具有突出的实质性特点，不具备创造性。

二、权利要求5、7、15、17不符合专利法实施细则第二十条第一款的规定。

(1)权利要求5与权利要求2都引用权利要求1，附加技术特征分别为“其中所述队列与入口端口相关联”、“其中所述队列与所述网络设备的入口端口相关联”，这里的“入口端口”实质上都是指“网络设备的入口端口”，因此，权利要求5与权利要求2的保护范围实质上相同，属于权利要求书不简要，不符合专利法实施细则第二十条第一款的规定；

基于相似理由，权利要求15与权利要求12二者的保护范围实质上也是相同的，使得权利要求书不简要。

(2)权利要求7和17中“Q号”不是本领域所熟知的技术术语，因此不能明确其具体含义，使得权利要求保护范围不清楚。

三、权利要求22请求保护一种计算机程序，其特征都是用计算机程序所能实现的功能限定，属于专利法第二十五条第一款第（二）项所述的智力活动的规则和方法的范围，不属于专利保护的客体。

四、独立权利要求29与独立权利要求1、11、23、26之间不具有单一性，不符合专利法第三十一条第一款的规定。

独立权利要求29请求保护一种网络设备中分配队列的方法，涉及确定入口端口可以接收的分组的第一数目，并给入口端口分配第二数目的物理队列及两个数目大小的比较；而独立权利要求1、11、23、26都涉及对分组进行分类并以此分配队列的方法或网络设备；虽然其都涉及在网络设备中分配队列，但独立权利要求29与独立权利要求1、11、23、26所要保护的技术方案之间没有相同或者相应的技术特征，更不具有相同或者相应的特定技术特征，因此其不具有单一性。

基于上述理由，本申请按照目前的文本还不能被授予专利权。申请人应按照本通知书提出的审查意见对申请文件进行修改，克服所存在的缺陷，否则本申请将被驳回。并且对申请文件的修改应当符合专利法第三十三条的规定，不得超出原说明书和权利要求书记载的范围。同时修改还应符合专利法实施细则第五十一条第三款的规定。

审查员：王澍
代码：953A

(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) EP 1 303 079 A2

(12) EUROPEAN PATENT APPLICATION

(43) Date of publication:
16.04.2003 Bulletin 2003/16

(51) Int Cl.7: H04L 12/24

(21) Application number: 02021812.9

(22) Date of filing: 27.09.2002

(84) Designated Contracting States:
AT BE BG CH CY CZ DE DK EE ES FI FR GB GR
IE IT LI LU MC NL PT SE SK TR
Designated Extension States:
AL LT LV MK RO SI

(30) Priority: 10.10.2001 US 328159 P

(71) Applicant: ALCATEL
75008 Paris (FR)

(72) Inventors:
- Morgan, David
Salt Lake City, Utah 84124 (US)
- See, Michael
Chapel Hill, NC 27514 (US)
- Clawson, Steve
Salt Lake City, Utah 84103 (US)

(74) Representative:
Dreiss, Fuhlendorf, Steimle & Becker
Patentanwälte,
Postfach 10 37 62
70032 Stuttgart (DE)

(54) Central policy based traffic management

(57) A switching node configuring different traffic management protocols via a single centralized set of policies. The switching node includes a central policy repository, a central policy engine, and a central management engine. The central policy repository stores a single set of policies used to manage a plurality of different traffic management protocols in a consistent and predictable manner. The different traffic management protocols may include QoS, NAT, ACL, and the like. The central policy engine evaluates inbound traffic flows based on the policies in the central policy repository, and configures one or more traffic management protocol entities based on a selected policy. The management engine configures and manages the single set of policies via a common set of commands helping to eliminate the danger of creating conflicting policies that may lead to unpredictable results.

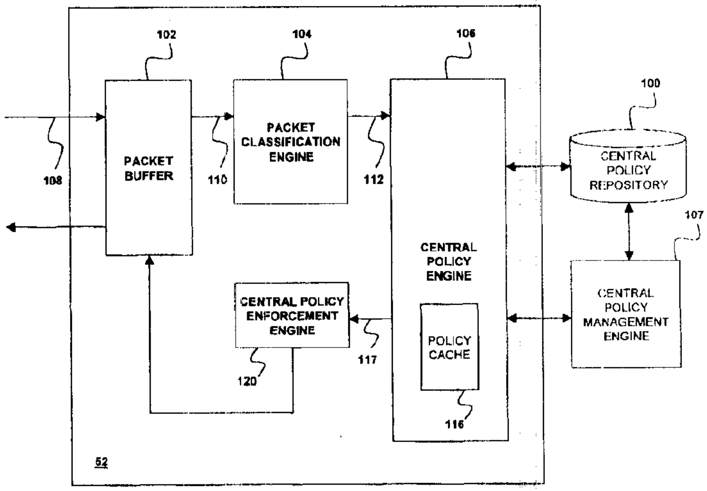

102
104
106
100
PACKET BUFFER
PACKET CLASSIFICATION ENGINE
108
110
112
CENTRAL POLICY REPOSITORY
107
CENTRAL POLICY ENGINE
CENTRAL POLICY ENFORCEMENT ENGINE
117
POLICY CACHE
CENTRAL POLICY MANAGEMENT ENGINE
120
116
52

FIG. 5

Printed by Jouve, 75001 PARIS (FR)

Description

CROSS-REFERENCE TO RELATED APPLICATION (S)

[0001] This application claims the benefit of U.S. provisional application No. 60/328,159, filed on October 10, 2001, the content of which is incorporated herein by reference.

FIELD OF THE INVENTION

[0002] The present invention relates generally to traffic control in a data communications network, and more particularly, to configuring different network protocols associated with traffic management via a single centralized set of policies.

BACKGROUND OF THE INVENTION

[0003] Policy based traffic control has become increasingly important in data communication networks as data traffic competes for limited bandwidth. Policy based traffic control generally involves separate, independent interfaces for configuring different traffic management protocols.

[0004] FIG. 1 is a schematic block diagram of a switching node 9 for policy based traffic control according to conventional mechanisms. The switching node includes separate, independent policy engines 11, 13, 15 and associated interfaces 17, 19, 21 for managing and configuring different traffic management protocols. For example, a switching node may include a separate policy engine for controlling quality of service (QoS), IP filtering via access control lists (ACL), network address translation (NAT), and the like. Each policy engine is associated with a control interface used by a network administrator to configure and manage the associated policies for a discrete traffic management protocol.

[0005] In general terms, an inbound packet is processed by a first policy engine, such as, a QoS policy engine, for a matching policy. If a match is found, the matching policy is enforced and the packet is generally not processed by the other policy engines. If a match is not found, the packet is processed by a next policy engine for a match. Under existing technology, therefore, multiple actions from different policy engines are either impossible or difficult to perform for a specific traffic flow. It is often desirable, however, to be able to perform such multiple actions. For example, it may be desirable to invoke a NAT policy engine for address translation based on a source IP address while simultaneously invoking the QoS policy engine for assigning a QoS priority to the flow based on the same source address.

[0006] Another problem with the current policy based traffic control is that the use of separate, independent interfaces for configuring the policies generally requires the network administrator to learn and use different command sequences for configuring and managing different policy engines. This may result in the configuration of conflicting policy rules that may lead to unpredictable results, especially if the configurations are done by different administrators or the same administrator at different times.

[0007] Accordingly, there is a need for a mechanism for providing and applying a common and consistent set of policies to traffic flowing through a data communications network. The mechanism should allow a switching node to enforce a singe policy with multiple actions in a consistent manner.

SUMMARY OF THE INVENTION

[0008] The present invention is directed to traffic management via a single centralized set of policies. According to one embodiment, the invention is directed to a switching node in a data communications network that includes an input, a repository, a policy engine, and a management engine. The repository stores a single set of policies for controlling a plurality of different traffic management protocols. The policy engine is coupled to the input and the repository, and is configured to evaluate an inbound packet based on a policy selected from the single set of policies, and configure one or more traffic management protocol entities based on the selected policy. The management engine is coupled to the repository and the policy engine. The management engine configures and manages the single set of policies via a common set of commands.

[0009] According to another embodiment, the invention is directed to a method for policy based traffic management where the method includes storing in a repository a single set of policies for controlling a plurality of different traffic management protocols. The method includes receiving a first packet, retrieving a first policy from the repository where the first policy identifies a first action for configuring a traffic management protocol entity of a first protocol type, and configuring the traffic management protocol entity based on the first action. The method also includes receiving a second packet, retrieving a second policy from the repository where the second policy identifies a second action for configuring a traffic management protocol entity of a second protocol type, and configuring the traffic management protocol entity based on the second action.

[0010] According to a further embodiment, the invention is directed to a method for policy based traffic management where the method includes storing in a repository a single set of policies for controlling a plurality of different traffic management protocols, receiving a packet, and retrieving a policy from the repository which identifies a first and second action for controlling traffic management protocol entities of a first and second protocol type. The method includes configuring the traffic management protocol entity of the first protocol type based on the first action and configuring the traffic management protocol of the second protocol type based on the second action.

[0011] In an additional embodiment, the invention is directed to a method for policy based traffic management where the method includes storing in a repository a single set of policies for controlling a plurality of different traffic management protocols, receiving a packet, and searching a policy cache for a policy applicable to the packet. If the policy cache does not include an applicable policy, the method includes searching the repository for a match, and generating and storing a new policy in the policy cache. The new policy is selected as applicable to a future packet if the repository contains no other policies that are also applicable to a future packet but are different from the new policy.

[0012] It should be appreciated, therefore, that the evaluation of traffic via a central policy engine allows the configuration of traffic management protocol entities of different protocol types, such as quality of service, access control, network address translation, and the like, in a consistent and predictable manner. The management of the policies via a common set of commands and the storage of the policies in a central repository help eliminate the creation and application of conflicting policies that could result in unpredictable results.

BRIEF DESCRIPTION OF THE DRAWINGS

[0013] These and other features, aspects and advantages of the present invention will be more fully understood when considered with respect to the following detailed description, appended claims, and accompanying drawings where:

FIG. 1 is a schematic block diagram of a switching node for policy based traffic management according to conventional mechanisms;

FIG. 2 is a schematic block diagram of a network environment including a packet switching node according to one embodiment of the invention;

FIG. 3 is a block diagram of a switching interface in one embodiment of the present invention;

FIG. 4 is a schematic block diagram of the packet switching controller providing and applying a common, centralized set of simple policies for controlling a plurality of traffic management protocols according to one embodiment of the invention;

FIG. 5 is a more detailed diagram of the switching controller of FIG. 4 according to one embodiment of the invention;

FIG. 6 is a conceptual layout diagram of various types of policy objects provided by a central management engine for creating a centralized set of policies according to one embodiment of the invention;

FIG. 7 is a conceptual layout diagram of a central policy repository according to one embodiment of the invention; and

FIG. 8 is a flow diagram of a central policy based traffic management according to one embodiment of the invention.

DETAILED DESCRIPTION

[0014] FIG. 2 is a schematic block diagram of a network environment including a packet switching node 10 according to one embodiment of the invention. The packet switching node may also be referred to as a switch, a data communication node or a data communication switch. The packet switching node 10 includes switching interfaces 14, 16 and 18 interconnected to respective groups of LANs 30, 32, 34, and interconnected to one another over data paths 20, 22, 24 via switching backplane 12. The switching backplane 12 preferably includes switching fabric. The switching interfaces may also be coupled to one another over control paths 26 and 28.

[0015] The switching interfaces 14, 16, 18 preferably forward packets to and from their respective groups of LANs 30, 32, 34 in accordance with one or more operative communication protocols, such as, for example, media access control (MAC) bridging and Internet Protocol (IP) routing. The switching node 10 is shown for illustrative purposes only. In practice, packet switching nodes may include more or less than three switching interfaces.

[0016] FIG. 3 is a block diagram of a switching interface 50 in one embodiment of the present invention. The switching interface 50 may be similar, for example, to the switching interfaces 14, 16, 18 of FIG. 2. The switching interface 50 includes an access controller 54 coupled between LANs and a packet switching controller 52. The access controller 54, which may, for example, include a media access controller (MAC), preferably receives inbound packets off LANs, performs flow-independent physical and MAC layer operations on the inbound packets and transmits the inbound packets to the packet switching controller 52 for flow-dependent processing. The access controller 54 also receives outbound packets from the packet switching controller 52 and transmits the packets on LANs. The access controller 54 may also perform physical and MAC layer operations on the outbound packets prior to transmitting them on LANs.

[0017] The packet switching controller 52 receives inbound packets, classifies the packets, modifies the packets in accordance with flow information and transmits the modified packets on switching backplane, such as the switching backplane 12 of FIG. 2. The packet switching controller 52 preferably also receives packets modified by other packet switching controllers via the switching backplane and transmits them to the access controller 54 for forwarding on LANs. The packet switching controller 52 may also subject selected ones of the packets to egress processing prior to transmitting them to the access controller 54 for forwarding on LANs.

[0018] FIG. 4 is a schematic block diagram of the packet switching controller 52 providing and applying a common, centralized set of simple policies for coordinating a plurality of traffic management protocols, such as, for example, access control, address translation, server load balancing, quality of service, and the like. The switching controller 52 includes a central policy engine 56 coupled to a central management engine 58. The central policy engine 56 evaluates the traffic flow against the centralized set of policies to perform, from a central location, one or more policy actions typically performed by separate, independent policy engines. The centralized set of policies include, but are not limited to system policies, network policies, access policies, services policies, and the like. The one or more actions include, but are not limited to packet filtering, packet prioritizing, address translation, server load balance group assignment, and assignment of 802.1p. TOS, or DSCP values.

[0019] The central management engine 58 may include software and/or hardware components to enable a network administrator to configure and manage the centralized set of policies. With the central management engine 58, the network administrator may, from a single location and using a common set of commands, create policies that manage different traffic management protocols.

[0020] FIG. 5 is a more detailed diagram of the switching controller 52 according to one embodiment of the invention. The switching controller includes a packet buffer 102, packet classification engine 104, central policy engine 106, central policy enforcement engine 120, central policy repository 100, and central management engine 107. The packet classification engine 104, central policy engine 106, central policy enforcement engine 120, and central policy management engine 107 are logical devices that may be implemented in software, hardware, firmware (e.g. ASIC), or any combination thereof. It is understood, of course, that FIG. 5 illustrates a block diagram of the packet switching controller without obfuscating inventive aspects of the present invention with additional elements and/or components that may be required for creating the controller. These additional elements and/or components, which are not shown in FIG. 5 are well known to those skilled in the art.

[0021] The switching controller 52 receives inbound packets 108. The packets may include, but are not limited to, Ethernet frames, ATM cells, TCP/IP and/or UDP/IP packets, and may also include other Layer 2 (Data Link/MAC Layer), Layer 3 (Network Layer) or Layer 4 (Transport Layer) data units.

[0022] The received packets are stored in the packet buffer 102. The packet buffer 102 provides via output signal 110 the stored packets or portions thereof to the packet classification engine 104 for processing.

[0023] The packet classification engine 104 may include one or more of a data extractor and a data cache. In an alternative embodiment, the data extractor and data cache are provided within the packet buffer 102.

[0024] The data extractor is used to extract one or more fields from the packets, and to store the extracted fields in the data cache as extracted data. The extracted data may include, but is not limited to, some or all of the packet header. For example, the extracted data may include, but are not limited to, one or more of Layer 2 MAC addresses, 802.1P/Q tag status, Layer 2 encapsulation type, Layer 3 protocol type, Layer 3 addresses, ToS (type of service) values, Layer 4 port numbers, portions of the packet body, and/or any other data used for determining a policy.

[0025] The extracted data is transmitted to the central policy engine 106 via an output signal 112. The central policy engine 106 may be similar to the central policy engine 56 of FIG. 4.

[0026] The central policy engine 106 accesses either an internal policy cache 116 or the central policy repository 100 for selecting a policy applicable to the packet. In accessing the central policy repository 100, the central policy engine 106 communicates with the repository using protocols such as, for example, LDAP.

[0027] According to one embodiment of the invention, the policy cache 116 includes sufficient information for applying policies to existing traffic flows without having to process the entire list of policies in the central policy repository 100 for every packet in the traffic flow. The information in the policy cache 116 is specific enough to prevent packets for which a different applicable rule may exist in the central policy repository 100 to be processed by the policy cache 116.

[0028] The central policy repository 100 may be implemented in a local memory and/or an external directory server with Lightweight Directory Access Protocol (LDAP) access. The central policy repository 100 includes a list of policies that are based on the contents of a packet and/or other elements such as, for example, time information, port information, and the like. In general terms, policies are rules composed of one or more conditions that describe a packet and one or more actions defining how the packet is to be processed if the condition is satisfied.

[0029] The central policy engine 106 compares the extracted packet data with either the policies in the policy cache 116 or central policy repository 100. If a match is found between the condition(s) in the policy and the extracted data, the policy engine determines the action (s) to be taken on the packet. The action(s) to be taken on the packet are transmitted to the central policy enforcement engine 120 via an output signal 117.

[0030] The central policy enforcement engine 120 ensures that the packet is processed according to the parameters defined in the action(s). In this regard, the central policy enforcement engine 120 interacts with other hardware and software elements in the switching node 30 for causing a desired processing of the packet. For example, if the policy actions specify that the packet is to be transmitted at a high priority, the policy enforcement engine 120 may direct the packet buffer 102 to place the packet in a high priority queue of an egress port.

[0031] The central management engine 107 of FIG. 5 may be similar to the central management engine 58 of FIG. 4. The engine 107 may be either a dedicated console or part of a network management console. A network administrator accesses the central management engine 107 for configuring and managing the policies in the central policy repository 100 and the central policy engine 106. According to one embodiment, the central management engine 107 provides a graphical user interface that provides a common set of commands and tools for configuring and managing polices that control different network elements such as, for example, QoS, NAT, ACL, and the like. The central management engine 107 also allows a network administrator to test policies before they are applied.

[0032] FIG. 6 is a conceptual layout diagram of various types of policy objects provided by the central management engine 107 for creating the centralized set of policies according to one embodiment of the invention. In the illustrated embodiment, the policy objects include rule 200 objects, condition 202 objects, action 204 objects, service 206 objects, and group 208 objects. Rules 200 are top level objects including conditions 202 and actions 204. According to one embodiment, rules are provided precedence values indicative of an order in which the rules are to be applied.

[0033] Conditions 202 are parameters used to classify traffic, and actions 204 are parameters describing how to treat the classified traffic. A condition 202 may include a service 206 and/or a group 208. A policy service 206 may be used as a shorthand for certain parts of a condition. According to one embodiment, a policy service 206 is defined by a service name, an IP protocol, a source IP port, and/or a destination IP port. For example, a "video" service may be defined as being associated with a "UDP" protocol and destination IP port number "4500." In another example, a "telnet" service may be defined as being associated with a "TCP" protocol and destination IP port number "23."

[0034] A policy group 208 may be defined by a list of IP/MAC addresses, ports, or services. Policy groups allow a network administrator to define conditions for a group of address, ports, or services, instead of creating a separate condition for each address, port, or service. For example, an "engineering" group may be defined as a set of particular IP addresses. A "basic services" group may be defined as a set of particular services such as telnet, FTP, HTTP, and Sendmail.

[0035] According to one embodiment of the invention, the central management engine 107 allows the creation of policies defining multiple actions for managing different traffic management protocols. For example, a policy in the central policy repository 100 may indicate that traffic with a particular source address and a particular destination address is to have the source address translated and receive a high priority. Thus, two different actions, a NAT policy action and a QoS policy action, may be defined via a single policy rule. FIG. 7 is a conceptual layout diagram of the central policy repository 100 according to one embodiment of the invention. In this illustrated embodiment, the repository includes a policy table 300 including a list of simple policy rules 302. Associated with each policy rule are a precedence number 304, condition 306, and action 308. The precedence number 304 indicates an order in which the rules are to be applied. If traffic matches more than one rule, the central policy engine 106 uses the rule with the highest precedence. In the illustrated embodiment, rule 4 is not matched since it is a subset, or more specific, than the higher precedence rule 2. The precedence ordering of the rules helps eliminate any rule conflicts and ensures that the results of evaluating a traffic flow against the policies is predictable and consistent.

[0036] The condition 306 for each rule defines parameters used for classifying inbound packets. These parameters include but are not limited to individual source addresses or source address groups, individual destination addresses or destination groups, and individual IP protocols 306e, individual IP ports 306d, or policy service groups 306c.

[0037] The action 308 for each rule defines one or more operations to be performed on the packet and/or traffic management protocol entities. The action 308 may be a filtering action, such as for example, dropping or admitting a packet. The action 308 may also be a QoS action such as, for example, assigning a priority to the packet. The action 308 may further be server load balancing, source or destination address translation, mapping or marking 802.1P, TOS, or DSCP values, or a combination of any of the discussed actions.

[0038] FIG. 8 is a flow diagram of a central policy based traffic control according to one embodiment of the invention. The process starts, and in step 400, the packet buffer 102 receives an inbound data packet and stores the packet in the buffer. In step 402, the packet classification engine 104 extracts one or more fields from the packets. In step 404, the central policy engine 106 determines whether the policy cache contains entries that match the extracted fields of the packet. If the answer in YES, the central policy enforcement engine 120 ensures that the policy action indicated in the matched entry of the policy cache is enforced.

[0039] If no match exists in the policy cache 116, the central policy engine 106 determines in step 408 whether there is an exact match of the extracted fields with conditions of a rule in the central policy repository 100. If the answer is YES, the central policy engine proceeds to program, in step 414, the policy cache 116 with the condition fields of the matched policy, the fields having the values of the corresponding extracted fields. This allows future data packets with the same extracted fields to match the newly programmed entry in the policy cache 116, avoiding another search of the central policy repository 100. In step 416, the central policy enforcement engine 120 proceeds to take the policy actions indicated in the matched policy rule.

[0040] If there is no exact match of the conditions of a rule in the central policy repository 100, a determination is made in step 410 whether a partial match exists. If the answer is YES, the central policy engine 106 proceeds to program the policy cache 116 in step 418 with the condition fields of the selected policy, the fields having the values of the corresponding extracted fields, and with a default action. In step 420, the central policy enforcement engine 120 proceeds to take the default policy action.

[0041] If the search of the central policy repository 100 does not result in even a partial match of the rules, the central policy engine 106 proceeds to program the policy cache 116, in step 412, with minimal information needed to forward the packet. According to one embodiment of the invention, such minimal information is a source address of the packet, a destination address of the packet, and a default policy action. In another embodiment of the invention, the minimal necessary information is simply the destination address and the default policy action. The default policy action is enforced by the central policy enforcement engine 120 in step 420.

[0042] The programming of the policy cache 116 will be best understood when considering the following example. Assume that the central policy repository 100 includes the rules depicted in FIG. 7.

[0043] A first packet received by the central policy engine 106 includes the following fields extracted by the classification engine 104:

Source IP - 192.200.200.200
Destination IP - 10.5.3.4
Protocol - TCP
Port - 80 (HTTP)

[0044] The central policy engine 106 compares the extracted information with entries in the policy cache 116. Assuming for purposes of this example that this is a first packet processed by the central policy engine 106, no entries are contained in the policy cache 116.

[0045] The central policy engine 106 then proceeds to search the central policy repository 100 for a match. Upon a finding of no match in the central policy repository 100, the central policy engine programs the policy cache with a minimal amount of information needed to process and forward future similar packets. In one example, the central policy engine may program the policy cache with a source IP address, destination IP address, and a default action. The default action in this example is the assignment of a default priority. The entry placed in the policy cache is as follows:

Source IP - 192.200.200.200
Destination IP - 10.5.3.4
Action - 0 (best effort priority)

[0046] The central policy engine 106 next receives a packet that includes the following fields:

Source IP - 192.200.200.200
Destination IP - 192.168.1.1
Protocol - TCP
Port - 80 (HTTP)

[0047] The central policy engine 106 compares the extracted information with entries in the policy cache 116. Upon a no match, the extracted information is compared against the rules in the central policy repository 100. Rule 1 matches the packet's source IP address, destination IP address, and protocol. However, there is no match in the port information, resulting in a partial match with rule 1.

[0048] In determining an entry for the policy cache, the central policy engine 106 ensures that the entry is specific enough to prevent packets for which a different applicable rule may exist in the central policy repository 100 to be processed by the policy cache. In this regard, the central policy engine 106 determines the number of condition fields of the rule that resulted in the partial match. Rule 1 that resulted in the partial match includes four condition fields: source IP address, destination IP address, protocol and port. Thus, the entry placed in the fast path includes four condition fields although only the source IP and the destination IP are needed to forward future packets. The value of the four fields is based on the information extracted from the packet. Accordingly, the entry placed in the policy cache is as follows:

Source IP - 192.200.200.200
Destination IP - 192.168.1.1
Protocol - TCP
Port - 80 (HTTP)
Action - 0 (best effort priority)

[0049] A next packet received by the central policy engine 106 includes the following fields:

Source IP - 192.200.200.200
Destination IP - 192.168.1.1
Protocol - TCP
Port - 21 (FTP)

[0050] The central policy engine 106 compares the extracted information with entries in the policy cache 116 and does not find a match. If, however, the policy cache had been programmed with less than four fields in processing the previous packet, a match would have resulted in the policy cache, causing the current packet to the forwarded without consulting the rules in the central policy repository.

[0051] The central policy engine 106 proceeds to search the central policy repository 100 and finds an exact match with rule 1. In determining the entry to be programmed in the policy cache, the central policy engine determines the number of condition fields in the rule that resulted in the exact match. The four condition fields of the matching rule 1 are then programmed into the policy cache. The value of the four fields is based on the information extracted from the packet. The entry placed in the policy cache is as follows:

Source IP - 192.200.200.200
Destination IP - 192.168.1.1
Protocol - TCP
Port - 21 (FTP)
Action - 7 (high priority)

[0052] The central policy engine 106 next receives a packet that includes the following fields:

Source IP - 192.200.200.200
Destination IP - 192.168.2.2
Protocol - UDP
Port - 300

[0053] The central policy engine 106 compares the extracted information with entries in the policy cache 116 and does not find a match. The central policy engine 106 then proceeds to search the rules in the central policy repository 100 and finds an exact match with Rule 2. The fields in Rule 2 that resulted in the exact match are source IP address and destination IP address. Accordingly, the entry placed in the policy cache is as follows:

Source IP - 192.200.200.200
Destination IP - 192.168.2.2
Action - 7 (high priority)

[0054] Although this invention has been described in certain specific embodiments, those skilled in the art will have no difficulty devising variations which in no way depart from the scope and spirit of the present invention. It is therefore to be understood that this invention may be practiced otherwise than is specifically described. Thus, the present embodiments of the invention should be considered in all respects as illustrative and not restrictive, the scope of the invention to be indicated by the appended claims and their equivalents rather than the foregoing description.

Claims

1. A switching node in a data communications network, the switching node comprising:

   an input receiving an inbound packet;
   a repository storing a single set of policies for controlling a plurality of different traffic management protocols;
   a policy engine coupled to the input and the repository, the policy engine evaluating the packet based on a policy selected from the single set of policies and configuring one or more traffic management protocol entities based on the selected policy; and
   a management engine coupled to the repository and the policy engine, the management engine configuring and managing the single set of policies via a common set of commands.

2. The switching node of claim 1, wherein the single set of policies includes a policy identifying two or more actions for controlling two or more traffic management protocols.

3. The switching node of claim 1, wherein the single set of policies includes a first policy identifying a first action for controlling a first traffic management protocol and a second policy identifying a second action for controlling a second traffic management protocol.

4. The switching node of claim 1, wherein one of the traffic management protocols is quality of service.

5. The switching node of claim 1, wherein one of the traffic management protocols is access control.

6. The switching node of claim 1, wherein one of the traffic management protocols is address translation.

7. The switching node of claim 1 further comprising a policy cache coupled to the repository and the policy engine for storing a plurality of cached policies.

8. The switching node of claim 7, wherein the policy engine is configured to evaluate the packet based on the plurality of cached policies prior to evaluating the packet based on the policies in the repository.

9. The switching node of claim 8, wherein the central policy engine selects a cached policy as applicable to the packet if no other policies are stored in the repository that are also applicable to the packet but are different from the selected cached policy.

10. The switching node of claim 8, wherein if no match of a policy is found in the repository, the policy engine is configured to store in the policy cache a policy having a destination address of the packet and a default action.

11. The switching node of claim 8, wherein if a partial match of a policy is found in the repository, the policy engine is configured to store in the policy cache a policy having condition fields of the policy in the repository where the values of the condition fields

are obtained from the packet, and a default action.

12. The switching node of claim 8, wherein if a complete match of a policy is found in the repository, the policy engine is configured to store in the policy cache a policy having condition fields of the policy in the repository where the values of the condition fields are obtained from the packet, and an action indicated by the policy in the repository.

13. A method for policy based traffic management comprising:

    storing in a repository a single set of policies for controlling a plurality of different traffic management protocols;
    receiving a first packet;
    retrieving a first policy from the repository, the first policy identifying a first action for configuring a traffic management protocol entity of a first protocol type;
    configuring the traffic management protocol of the first protocol type based on the first action;
    receiving a second packet;
    retrieving a second policy from the repository, the second policy identifying a second action for configuring a traffic management protocol entity of a second protocol type; and
    configuring the traffic management protocol entity of the second protocol type based on the second action.

14. A method for policy based traffic management comprising:

    storing in a repository a single set of policies for controlling a plurality of different traffic management protocols;
    receiving a packet;
    retrieving a policy from the repository, the policy identifying a first and second action for controlling a first and second traffic management protocol entity, respectively, of a first and second protocol type, respectively;
    configuring the first traffic management protocol entity based on the first action; and
    configuring the second traffic management protocol entity based on the second action.

15. A method for policy based traffic management comprising:

    storing in a repository a single set of policies for controlling a plurality of different traffic management protocols;
    receiving a packet;
    searching a policy cache for a policy applicable to the packet;
    searching the repository if the policy cache does not include an applicable policy; and
    generating and storing a new policy in the policy cache,

    wherein if the new policy is selected as applicable to a future packet, no policies are stored in the repository that are also applicable to the future packet but are different from the new policy.

16. The method of claim 15, wherein if no match of a policy is found in the repository, the policy generated and stored in the policy cache includes a destination address of the packet and a default action.

17. The method of claim 15, wherein if a partial match of a policy is found in the repository, the policy generated and stored in the policy cache includes condition fields of the policy in the repository where the values of the condition fields are obtained from the packet, and a default action.

18. The method of claim 15, wherein if a complete match of a policy is found in the repository, the policy generated and stored in the policy cache includes condition fields of the policy in the repository where the values of the condition fields are obtained from the packet, and an action indicated by the policy in the repository.

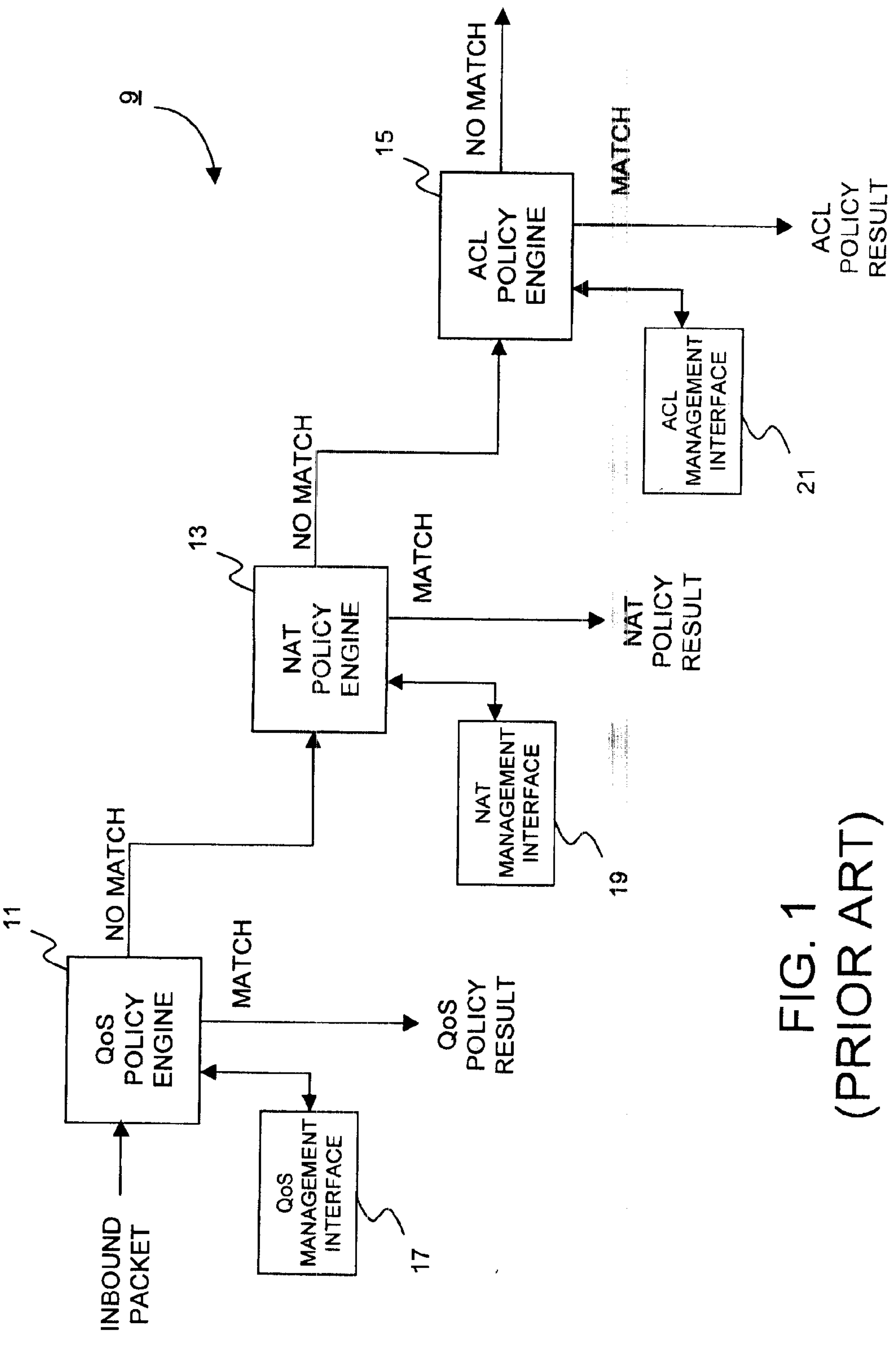


FIG. 1
(PRIOR ART)

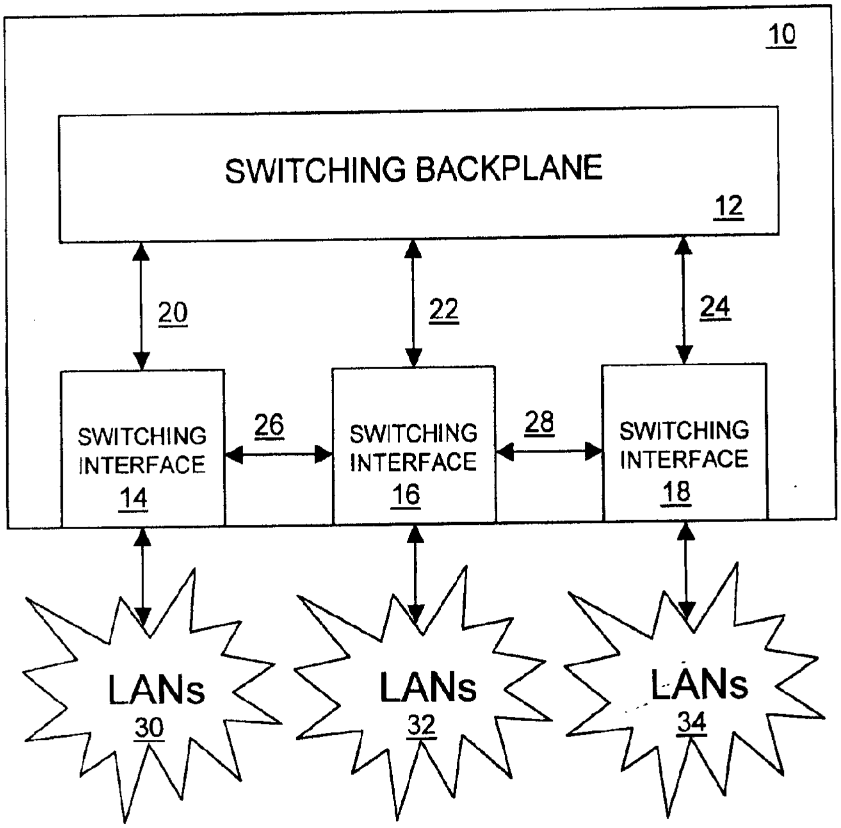

10
SWITCHING BACKPLANE
12
20
22
24
SWITCHING INTERFACE 14
26
SWITCHING INTERFACE 16
28
SWITCHING INTERFACE 18
LANs 30
LANs 32
LANs 34

FIG. 2

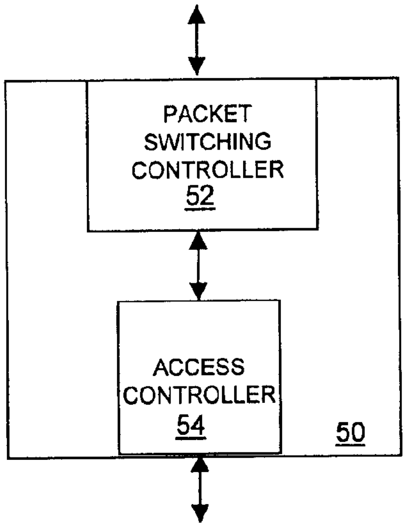

PACKET SWITCHING CONTROLLER 52
ACCESS CONTROLLER 54
50

FIG. 3

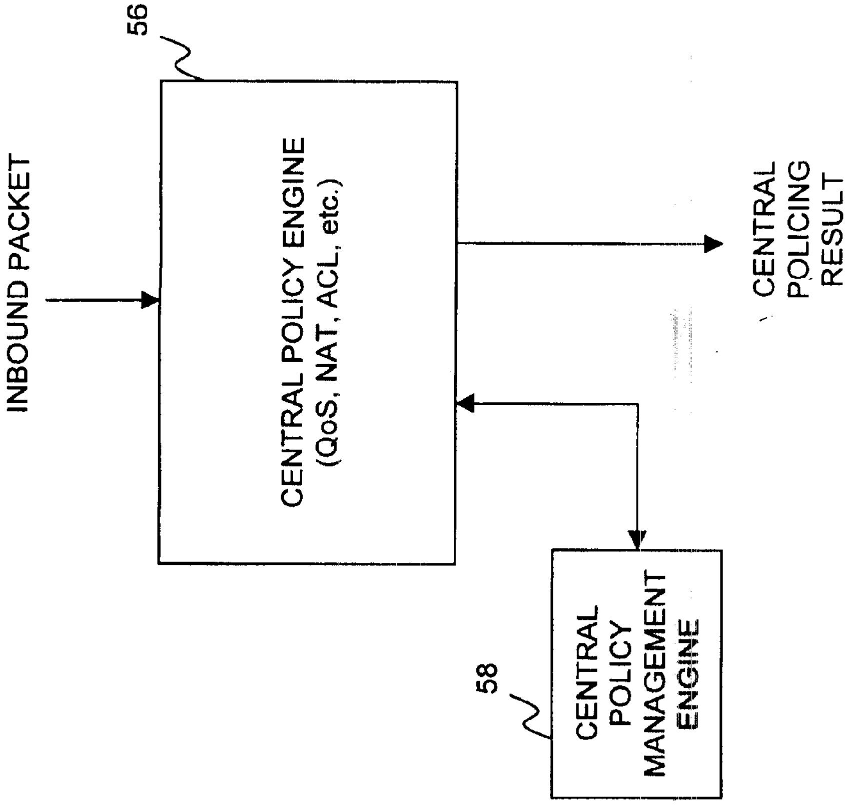

INBOUND PACKET
56
CENTRAL POLICY ENGINE
(QoS, NAT, ACL, etc.)
CENTRAL POLICING RESULT
52
58
CENTRAL POLICY MANAGEMENT ENGINE

FIG. 4

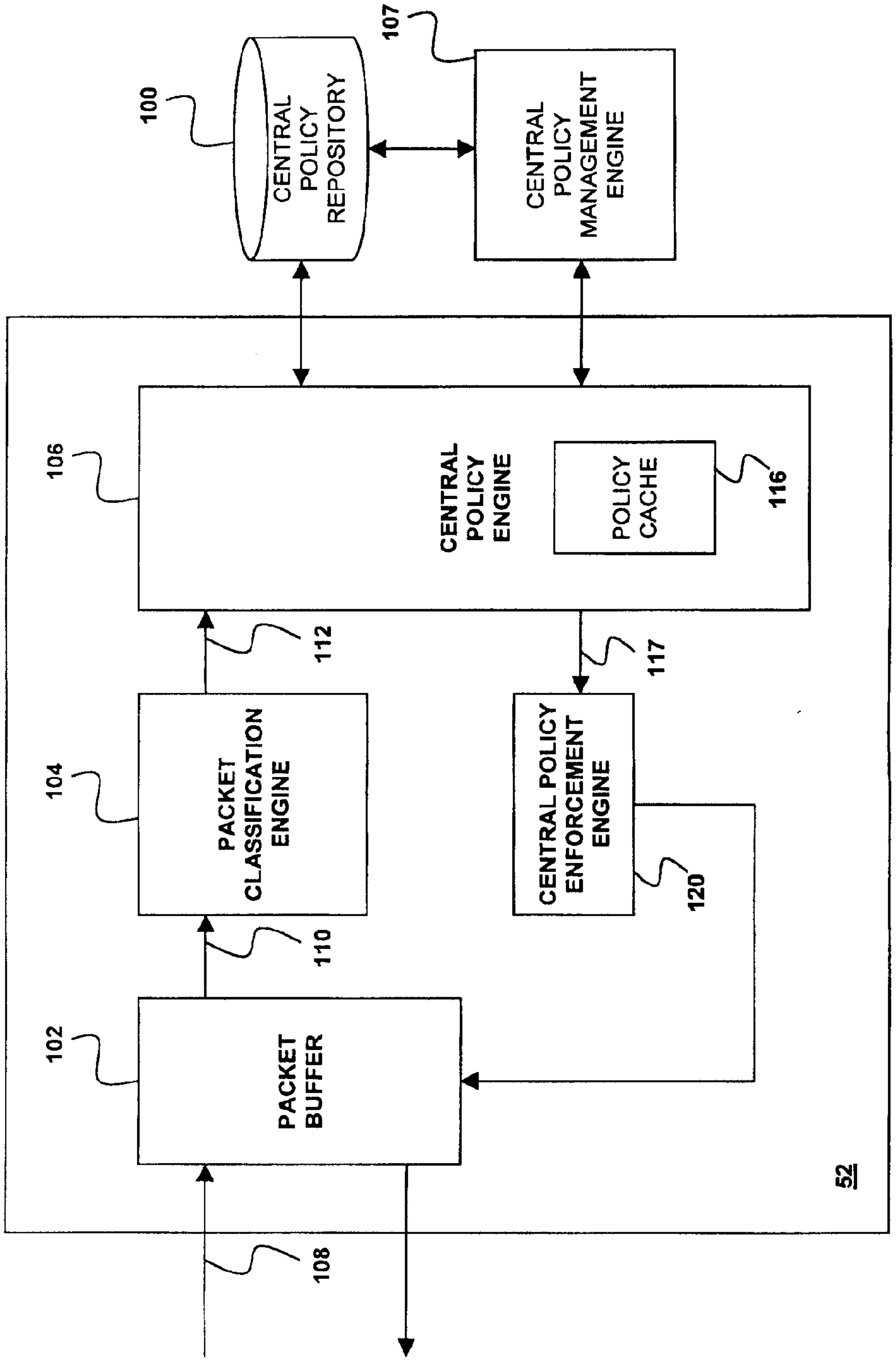

100
CENTRAL POLICY REPOSITORY
107
CENTRAL POLICY MANAGEMENT ENGINE
106
CENTRAL POLICY ENGINE
POLICY CACHE
116
112
117
104
PACKET CLASSIFICATION ENGINE
CENTRAL POLICY ENFORCEMENT ENGINE
120
110
102
PACKET BUFFER
108
52

FIG. 5

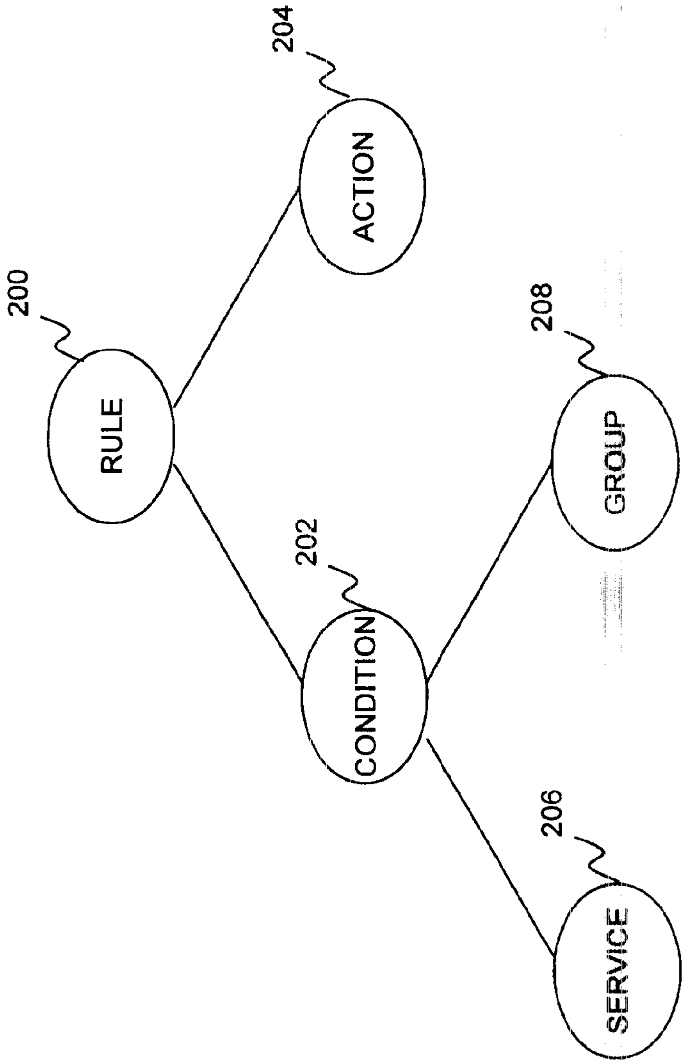
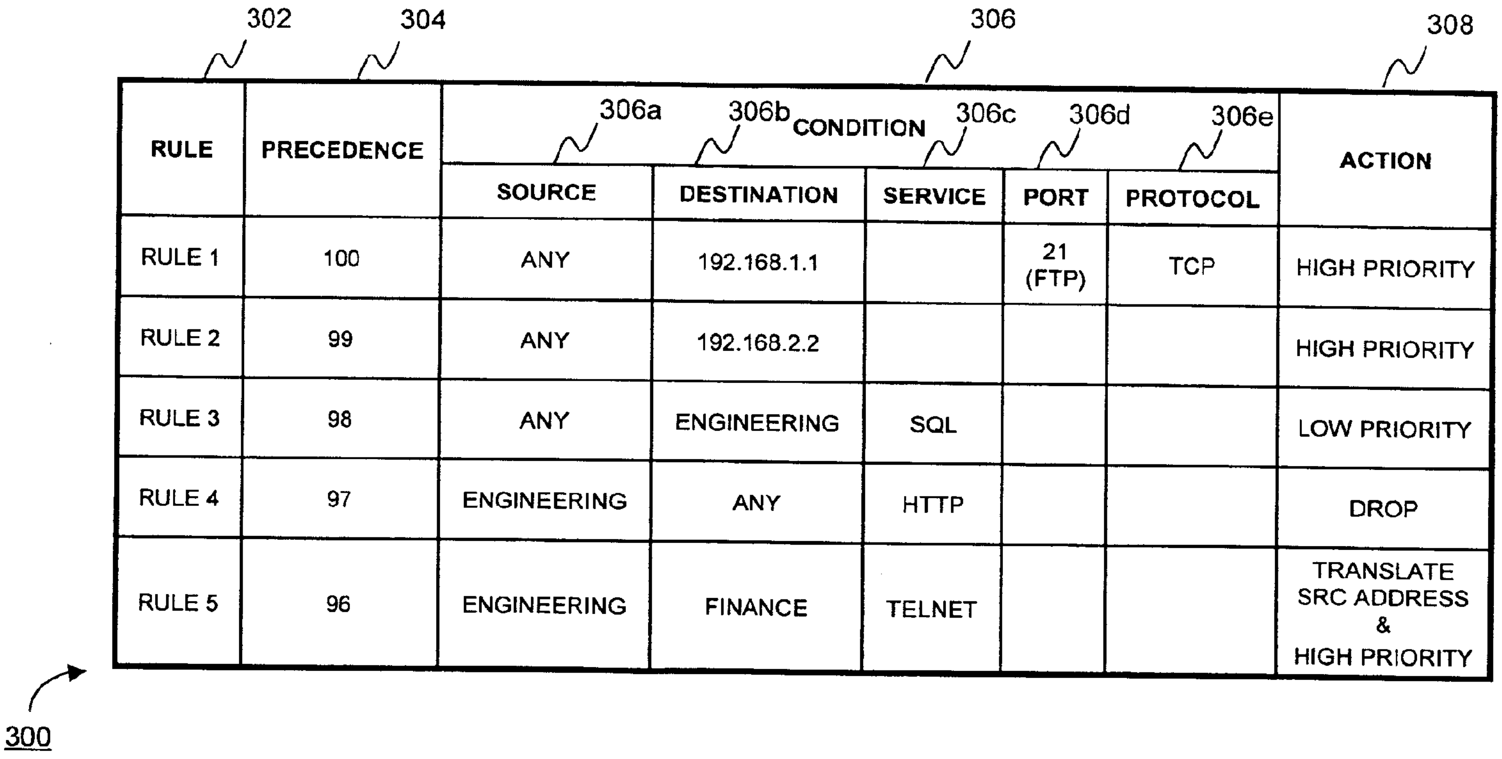


FIG. 6



302
304
306
308
306a
306b
306c
306d
306e
300

RULE	PRECEDENCE	CONDITION					ACTION
		SOURCE	DESTINATION	SERVICE	PORT	PROTOCOL	
RULE 1	100	ANY	192.168.1.1		21 (FTP)	TCP	HIGH PRIORITY
RULE 2	99	ANY	192.168.2.2				HIGH PRIORITY
RULE 3	98	ANY	ENGINEERING	SQL			LOW PRIORITY
RULE 4	97	ENGINEERING	ANY	HTTP			DROP
RULE 5	96	ENGINEERING	FINANCE	TELNET			TRANSLATE SRC ADDRESS & HIGH PRIORITY

FIG. 7

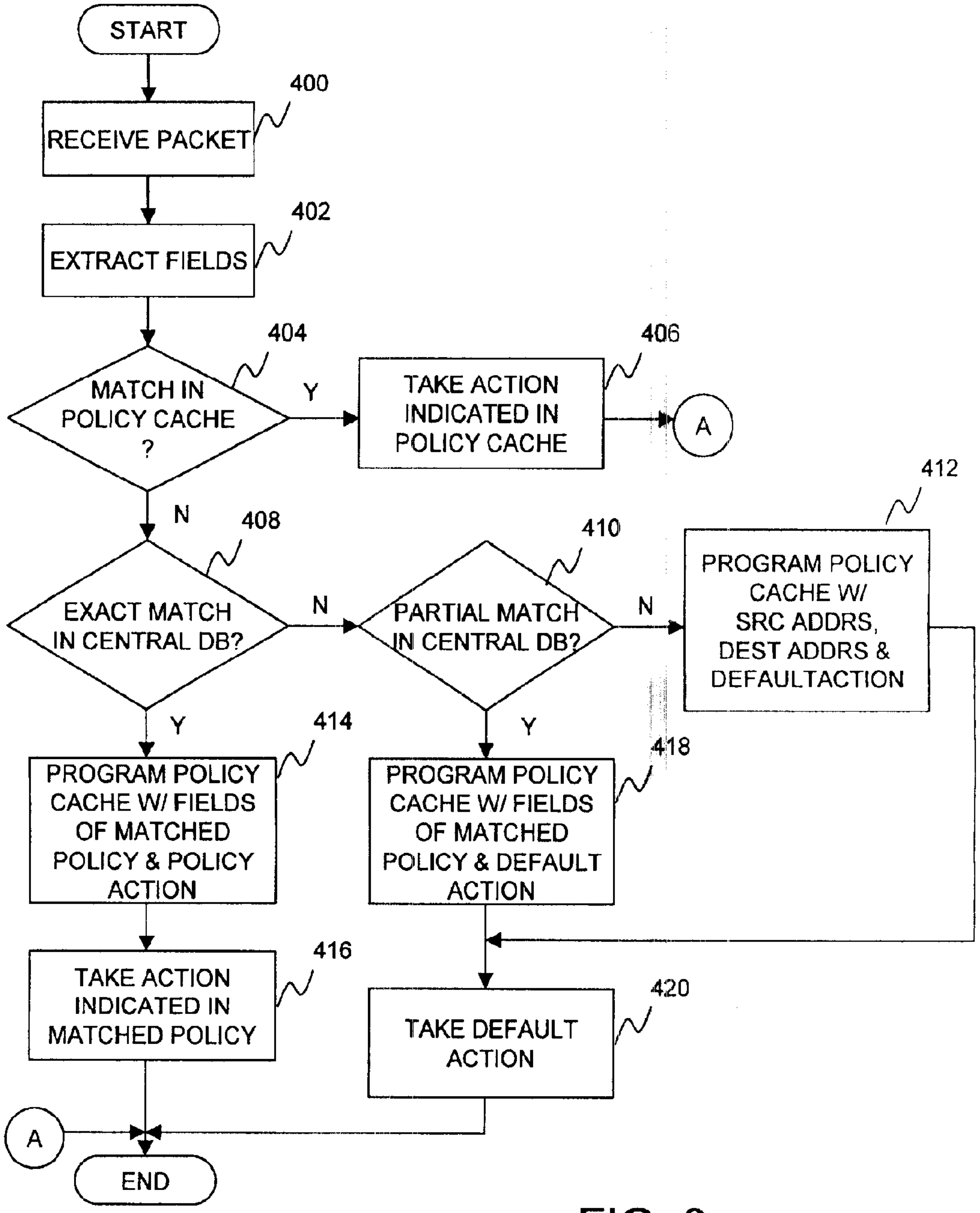

START
400
RECEIVE PACKET
402
EXTRACT FIELDS
404
MATCH IN POLICY CACHE ?
Y
406
TAKE ACTION INDICATED IN POLICY CACHE
A
N
408
EXACT MATCH IN CENTRAL DB?
N
410
PARTIAL MATCH IN CENTRAL DB?
N
412
PROGRAM POLICY CACHE W/ SRC ADDRS, DEST ADDRS & DEFAULTACTION
Y
414
PROGRAM POLICY CACHE W/ FIELDS OF MATCHED POLICY & POLICY ACTION
Y
418
PROGRAM POLICY CACHE W/ FIELDS OF MATCHED POLICY & DEFAULT ACTION
416
TAKE ACTION INDICATED IN MATCHED POLICY
420
TAKE DEFAULT ACTION
A
END

FIG. 8

[19]中华人民共和国国家知识产权局

[51] Int. Cl^7
H04Q 3/64
H04Q 3/52 H04B 10/12
H04B 10/24 H04B 10/08

[12]发明专利申请公开说明书

[21]申请号 01143804.5

[43]公开日 2002年7月17日

[11]公开号 CN 1359241A

[22]申请日 2001.12.13 [21]申请号 01143804.5
[30]优先权
[32]2000.12.14 [33]EP [31]00311184.6
[71]申请人 朗迅科技公司
地址 美国新泽西州
[72]发明人 杰瑞恩·沃伦

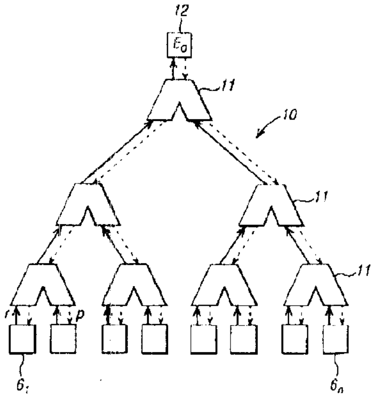
[74]专利代理机构 中国国际贸易促进委员会专利商标事务所
代理人 蒋世迅

权利要求书3页 说明书10页 附图页数6页

[54]发明名称 用于分组交换机和无源光网络的分布式调度器

[57]摘要

一种调度系统和方法，用于从输入端口$(1_1 \cdots 1_i)$调度数据分组到输 出端口$(3_1 \cdots 3_0)$，它包括虚拟输出队列$(6_1 \cdots 6_n)$，虚拟输出队列$(6_1 \cdots 6_n)$安排成存储来自输入端口$(1_1 \cdots 1_i)$的数据分组，其目的地是一个特 定的输出端口$(3_1 \cdots 3_0)$。调度系统包括有多个比较层的调度树(10)，每个比较层安排成成对比较从相关虚拟输出队列$(6_1 \cdots 6_n)$并行接收的请 求，以及在剩下单个请求之前，发送较高优先级请求到较高级比较层，单个请求指出被调度的虚拟输出队列$(6_1 \cdots 6_n)$发送它的数据分组到相关的输出端口$(3_1 \cdots 3_0)$。



知识产权出版社出版

说　明　书

用于分组交换机和无源光网络的分布式调度器

技术领域

本发明涉及用于分组交换机的调度器，具体涉及从多个输入端口调度数据分组到至少一个输出端口的方法，该方法包括以下步骤：在多个虚拟输出队列中存储数据分组，虚拟输出队列安排成存储来自多个输入端口中一个输入端口的数据分组，其目的地是该至少一个输出端口中特定的一个输出端口，以及调度多个虚拟输出队列。

背景技术

在太拉比特交换机和吉比特无源光网络（PON）中调度分组要求相当大的计算功率量。若必须部署优先级机构以管理不同服务质量（QoS）的业务，则问题将变得更加复杂。这种复杂性可以表示成系统中每个输出端口需要调度的输入队列总数，即，输入端口数目与服务等级数目的乘积。需要有这样一种算法，能够按照它们具体的优先级调度大量队列中的分组。在最新技术中，即，ASIC 或 FPGA，必须有效地执行这种算法。

C.Blondia，O.Casals 和 J.Garcia 的文章：‘A Cell Based MAC Protocol with Traffic Shaping and a Global FIFO Strategy’，Proceedings of the RACE Open Workshop on Broadband Access，Nijmegen，The Netherlands，June 1993，公开一种媒体接入协议，它利用部署普通的先进先出（FIFO）缓冲器的请求/准许机构。每个网络终端（NT）通过请求广告它的带宽要求，包括 NT 中队列状态信息。利用带宽分配算法，媒体接入协议分配可用的带宽给各个 NT。借助于准许通知 NT 有关分配的带宽。这种用于 PON 的算法（具体地说，异步转移方式（ATM）PON）只寻址少量的队列（~64 个队列），不适合于吉比特容量的大系统（~1000 个队列）。此外，还要求连接 PON 到核心网的附加交换功能。

I.Elhanany，J Nir，D.Sadot 的文章：‘A Contention-Free Packet

Scheduling Scheme for Provision of Quality-of-Service in Tbit/sec WDM Networks'，Optical Networks Magazine, July 2000，公开一种分组交换机的调度方案。提出的算法声称每个分组时隙周期约 N*^{2}log（N）次操作，其中 N 是输出端口或目的地数目（该文章涉及 N x N 交换机）。利用循环过程，包括每个输入端口的优先匹配方案以符合各种服务质量的要求，这种方法采用顺序断言不同的输入端口。对于大量的队列，这种方法仍然太慢。它还不能寻址 PON.

发明内容

本发明试图提供一种用于分组交换机和 PON 的调度器，它能够按照它们特定优先等级调度大量队列中的数据分组。队列的数目等于输入端口的数目，或在管理有不同服务质量要求的数据业务情况下，队列的数目等于输入端口数目与服务等级（或优先等级）数目的乘积。

本发明提供一种按照上述前序部分的方法，其中调度多个虚拟输出队列的步骤包括：借助于调度树，调度与至少一个输出端口中一个端口相关的虚拟输出队列，从而并行地调度与该至少一个输出端口中一个端口相关的虚拟输出队列，调度树至少包括一个比较层，用于执行成对比较从相关虚拟输出队列并行接收的请求，以及在剩下单个请求之前，发送较高优先级请求到较高级比较层，单个请求指出被调度的虚拟输出队列发送它的数据分组到相关的输出端口。

按照本发明的方法有以下的优点，可以有效地调度非常大量的虚拟输出队列。本发明的调度方法只需要 ^{2} log N 次操作，其中 N 是虚拟输出队列的数目。通过级联式接入共享媒体和接入输出端口，该方法不但可以有效地用于分组交换机，还可以用于无源光网络。在相关的分组交换机或无源光网络中所有输出端口，可以并行地执行本发明方法。

在本发明方法的一个实施例中，请求包括识别相关的虚拟输出队列。它允许直接识别准许接入到某个输出端口的虚拟输出队列。

在另一个实施例中，比较层还执行存储较高优先级请求的步骤，在从较高级接收到包括单个请求的准许之后，按照与较高优先级相关的存储请求，发送该准许到较低级比较层。这个实施例可以简化分配机构，避

[19]中华人民共和国专利局

[11] 公开号 CN 1099923A

[12]发明专利申请公开说明书

[21]申请号 94102225.0

[51]Int.Cl^5

H04Q 3／64

[43]公开日 1995年3月8日

[22]申请日 94.3.3

[30]优先权

[32]93.3.3 [33]US[31]08／025,538

[71]申请人 罗姆公司

地址 美国加利福尼亚州

[72]发明人 马克·卡明斯基 罗伯特·佩雷尔曼

彼平·佩特尔 珍妮·伊卡诺斯基

克里斯·袁

[74]专利代理机构 柳沈知识产权律师事务所

代理人 吴秉芬

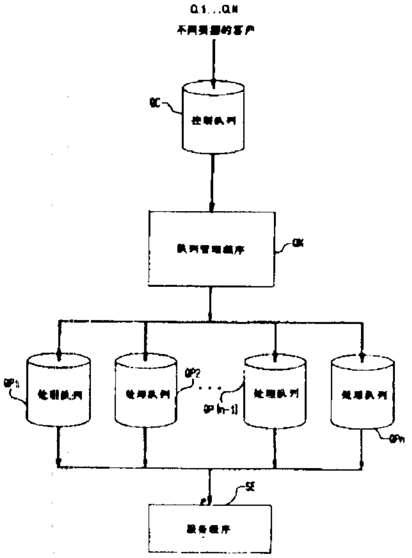
H04M 3／42

说明书页数: 附图页数:

[54]发明名称 队列管理系统和方法

[57]摘要

对多个具有不同客户类型的客户服务的队列管理系统，控制队列以预置顺序对客户进行排列。队列管理程序分配和再分配少于客户类型数的多个处理队列，以与各种客户类型相适应。如果有一相适应的处理队列时该队列管理程序连续地将客户安置该处理队列中，如果不存在相适应的处理队列但有一空的处理队列时该队列管理程序分配或再分配该空的或被空出的处理队列给该客户类型。上述方法可用于语音邮政电话系统的环境中。



为下一个被传送的客户类型。

由于该队列管理程序总是将处理队列已经包含的同一客户类型增补到每个处理队列，该队列管理程序将每个处理队列分配给该队列管理程序首先安置的该处理队列的客户类型。这种分配连续进行直到该服务程序使该处理队列空出为止。如果该队列管理程序现在在该空出的处理队列中安置一新的客户类型，那么该队列管理程序将解除处理队列对前面的客户类型的分配并将它再分配给最新的类型。解除分配和再分配容许一有限的处理队列数目（例如50）去处理任何数量的客户类型（例如1000或更多）。

根据本发明的一特殊方面，即在一电话环境中的条件下，所谓客户是为传送到多个终端而编码的记录语音邮政信息。对于所有终端的所有信息以一预置的顺序出现在该控制队列中，例如以一先入先出为基础。在比其终端数目要少的多个处理队列之间，队列管理程序通过将控制队列中的每个相继的第一信息安置到其终端与该第一信息的终端相适应的处理队列中的方法，一个接一个地分配该控制队列中的信息。如果不存在有相适应的处理队列，但存在一空出的处理队列时，则该队列管理程序将该第一信息置入该空出的处理队列。如果不存在空出的处理队列，则该队列管理程序在控制队列中保持该信息直至所有的处理队列的信息传送而使一处理队列空出为止。该队列管理程序这时将该处理队列再分配仅用来传送到新的终端的信息并将下一个信息安置到该空出的处理队列。

本发明使用了有限数量的队列来提供同时对大量的客户或终端的服务。实际上对不同客户类型或终端并设有限制。用于同一终端的同一类型的客户或信息是在同一队列中等待的。在同一队列中等

待的客户或信息，虽然可能有不同的排序，但在正常状态下是以年月日顺序排列的。对不同客户类型或信息终端的同时处理可容易实现。该系统动态地将队列分配给每个客户类型或终端，并当它们变为空出以及需用新终端的新的客户类型或信息到来时将它们重新分配。为了处理容易起见，本发明将与一给定客户类型或终端相关的处理与相应的队列关联起来。

对于多个客户类型或用于多个终端的信息的处理比用于一单一队列方案及对上述多个队列系统的处理变得更为有效。本发明考虑到在处理中有大的适应性。

本发明的这些和其它的特征在构成本说明书的一部分的权利要求中指出。本发明的其它目的和优点为借助于附图阅读时对本领域的技术人员来说将变得显然。

图1是一说明体现本发明特征的队列管理系统的方框图；

图2是说明在图1中所示的并体现了本发明的队列管理系统的操作流程图；

图3是一说明在图1中所示的并体现了本发明的队列管理系统的操作的另一种方法的流程图；

图4是一说明当它们开始图1所示的处理队列的服务时该服务程序的操作的流程图；

图5是说明当它们进行在该队列中的操作时在图1中所示的服务程序的操作的另外的流程图；

图6是一体现本发明的一实施例的一电话系统的方框图；

图7是在图6所示的电话系统的环境中体现本发明的一队列管理系统的一方框图；

US005920568A

United States Patent [19]

Kurita et al.

[11] **Patent Number:** **5,920,568**

[45] **Date of Patent:** **Jul. 6, 1999**

[54] **SCHEDULING APPARATUS AND SCHEDULING METHOD**

[75] Inventors: **Toshihiko Kurita**; **Ichiro Iida**, both of Kanagawa, Japan

[73] Assignee: **Fujitsu Limited**, Kanagwa, Japan

[21] Appl. No.: **08/790,443**

[22] Filed: **Jan. 29, 1997**

[30] **Foreign Application Priority Data**

Jun. 17, 1996 [JP] Japan 8-155746

[51] **Int. Cl.**^{6} **H04L 12/28**; H04L 12/56

[52] **U.S. Cl.** **370/412**; 370/411; 370/429

[58] **Field of Search** 370/414, 416–418, 370/395, 468, 229, 335, 351, 428, 429, 411, 412

[56] **References Cited**

U.S. PATENT DOCUMENTS

5,231,633	7/1993	Hluchyj et al.	370/94
5,268,900	12/1993	Hluchyj et al.	370/94
5,499,238	3/1996	Shon	370/411

Primary Examiner—Chi H. Pham
Assistant Examiner—Afsar M. Qureshi
Attorney, Agent, or Firm—Helfgott & Karas, P.C.

[57] **ABSTRACT**

A scheduling apparatus and a scheduling method are capable of reading data elements from a plurality of queues in such a form that past hysteresis reflects therein. The scheduling apparatus comprises a queue hysteresis table for storing a value e_count obtained subtracting the number of data elements (packets in a router) actually fetched out of the queue, from the number of times with which this queue becomes a processing target with respect to each queue. The apparatus also comprises a scheduling unit for cyclically designating each queue as a processing target, adding "1" to e_count, corresponding to that queue, in the queue hysteresis table if no data elements exist in the queue designated as the processing target, consecutively fetching, from the processing target queue, the data elements the number of which corresponds to a value of e_count corresponding to the queue if the data elements exist in the processing target queue, and decrementing the value of e_count by the number of fetched data elements.

10 Claims, 13 Drawing Sheets

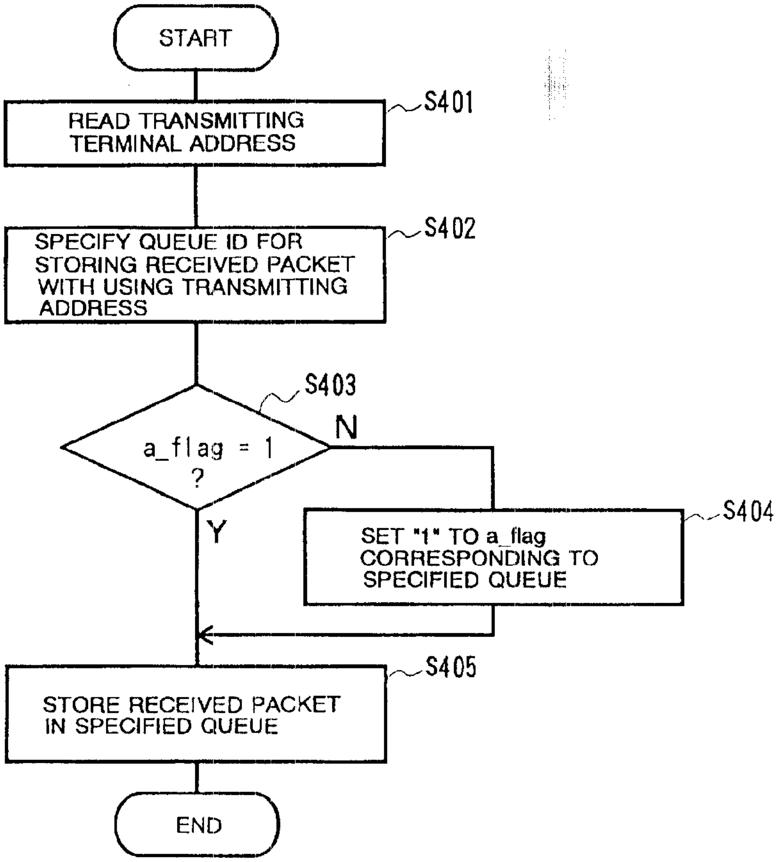

START
READ TRANSMITTING TERMINAL ADDRESS
S401
SPECIFY QUEUE ID FOR STORING RECEIVED PACKET WITH USING TRANSMITTING ADDRESS
S402
S403
a_flag = 1 ?
N
Y
SET "1" TO a_flag CORRESPONDING TO SPECIFIED QUEUE
S404
STORE RECEIVED PACKET IN SPECIFIED QUEUE
S405
END

FIG. 1

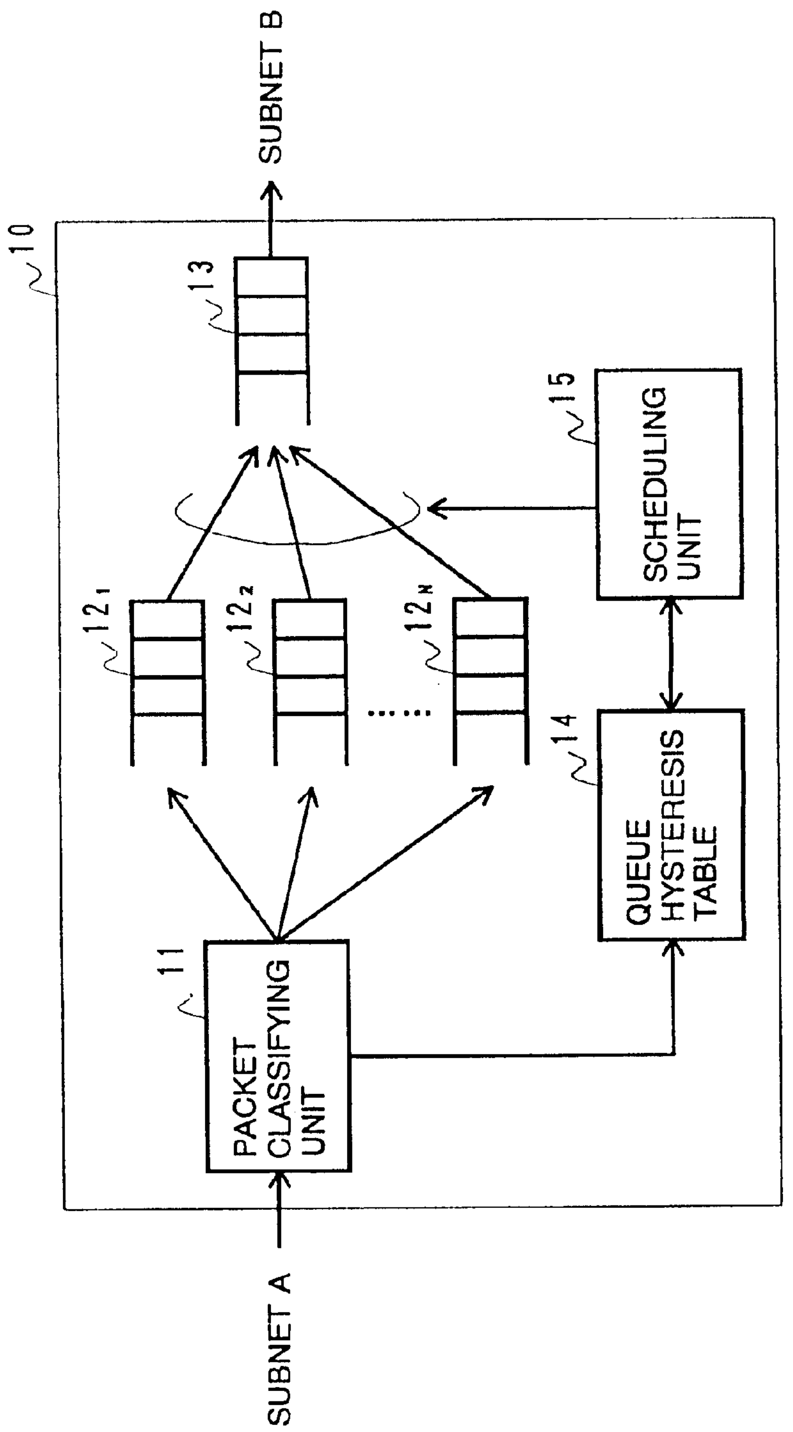

SUBNET A
SUBNET B
10
11
PACKET CLASSIFYING UNIT
12₁
12₂
12ɴ
13
14
QUEUE HYSTERESIS TABLE
15
SCHEDULING UNIT

FIG. **8** is an explanatory diagram showing an outline of the queue hysteresis table incorporated into the router in a second embodiment;

FIG. **9** is a flowchart showing operation procedures of the scheduling unit provided in the router in the second embodiment;

FIG. **10** is an explanatory diagram showing an outline of the queue hysteresis table incorporated into the router in a third embodiment;

FIG. **11** is a flowchart showing operation procedures of the scheduling unit provided in the router in the third embodiment;

FIG. **12** is a diagram illustrating an example of internet working that uses the router;

FIG. **13** is a block diagram showing functions of a router employing a prior art fair queuing method; and

FIG. **14** is an explanatory diagram showing scheduling procedures by the router using the prior art fair queuing method.

DESCRIPTION OF THE PREFERRED EMBODIMENTS

The present invention will hereinafter be specifically described with reference to the accompanying drawings.

First Embodiment

FIG. **1** illustrates an outline of a configuration of a router to which a scheduling method (and a scheduling apparatus) of the present invention is applied. A router **10** in a first embodiment is an apparatus for connecting a subnet A to a subnet B, and includes, as shown in FIG. **1**, a packet classifying unit **11**, queues 12_1–12_N, and output buffer **13**, a queue hysteresis table **14** and a scheduling unit **15**. Note that, the router **10** is constructed so that some queues **12** are created in a memory dynamically in accordance with a receiving condition of packets. Therefore, queues 12_1–12_N are not always exist in the router **10**. Although in this embodiment, assuming, for the sake of convenience, that the router **10** is provided with N queues **12** which functions independently, the configuration and operation will be discussed first. Thereafter, real configuration and operation of the router **10** will be discussed.

The queues 12_1–12_N are buffers for temporarily storing packets that should be transmitted to the subnet B, and are each made corresponding to transmitting stations (terminals) connected to the subnet A. The packet classifying unit **11** supplies the queues **12** corresponding to transmitting addresses contained in those packets with packets received from the subnet A. Note that the router **10** in this embodiment is constructed so as to transfer packets (so-called internet packet) each having a structure shown in FIG. **2**. Hence, the packet classifying unit **11** specifies a queue **12** to store the received packet by reading the transmitting terminal address (transmitting terminal IP address) included in the header of the received packet.

Further, the packet classifying unit **11** executes a process of rewriting contents of the queue hysteresis table **14** (the details of which will be stated later on) in parallel to the above-described supplying process of the packets. The output buffer **13** is temporarily stored with the packets within the respective queues **12**, and the packets in the output buffer **13** are transmitted in the as-inputted sequence to the subnet B.

The queue hysteresis table **14** is a table the contents of which are updated by the packet classifying unit **11** and the scheduling unit **15**. As illustrated in FIG. **3**, the queue

hysteresis table **14** is stored with active flags (a_flag) and empty count numbers (e_count) in such a form that these flags and numbers are made corresponding to queue IDs defined as data for identifying the queues.

The packet classifying unit **11** rewrites values of the active flags a_flag within the queue hysteresis table **14** in accordance with a packet receiving condition. More specifically, when receiving a packet, the packet classifying unit 11 reads, as shown in FIG. **4**, a transmitting terminal address included in the header of the received packet first (step **S401**). Next, the packet classifying unit **11** specifies a queue ID for storing the received packet by referring to a address-queue ID table which holds the relationship between the transmitting terminal addresses and the queue IDs, and is provided inside the unit (step **S402**). Thereafter, the packet classifying unit **11** checks the value of the active flag corresponding to the specified queue ID in the queue hysteresis table **14** (step **S403**).

When the value of the active flag is "0" (step **S403**;N), the packet classifying unit **11** rewrites the active flag to "1" (step **S404**). Then, the packet classifying unit **11** carries out a process for storing the received packet in the specified queue **12** (step **S405**). On the contrary, when the active flag is "1" (step **S403**;Y), the packet classifying unit **11** carries out a process for storing the received packet in the specified queue **12** (step **S405**) without rewriting the active flag.

Moreover, the packet classifying unit **11**, parallel to (independently on) the series of the above described processes, performs a process for managing receiving time of the last received packet for each queue ID the active flag of which is set to "1". Then, the packet classifying unit **11** changes, based on the managing results, the active flag concerning the queue ID of which the receiving time becomes before a time which a predetermined time (such as 60 sec.) is subtracted from a current time to "0".

The scheduling unit **15** performs control to transmitting the packets stored in the queues **12** to the output buffer **13**. The scheduling unit **15**, when executing this control, refers to values of the flags a_flag in the queue hysteresis table **14**, and refers to and updates values of the empty count numbers e_count.

Operations of the router and the scheduling unit **15** in this embodiment will hereinafter be described specifically.

FIG. **5** is a flowchart showing operation procedures of the scheduling unit **15** after starting up the router. As shown in FIG. **5**, the scheduling unit **15**, when actuating this router, to begin with, sets "1" in variables i and j respectively, and initializes the contents of the queue hysteresis table **14** (step **S101**). In step **S101**, the scheduling unit **15** sets "0" in all of the flags a_flag and of the empty count numbers e_count, thereby initializing the queue hysteresis table **14**. Moreover, the packet classifying unit **11**, after the scheduling unit **15** has executed step **S101**, starts executing the above-mentioned processes (of supplying the respective queues with the packets and updating the values of a_flag in the queue hysteresis table **14**).

After initializing the queue hysteresis table **14**, the scheduling unit **15** judges whether an active flag a_flag, of the queue the queue ID of which is i, is "1" or not (step **S102**). If a_flagis not "1", (step **S102**; N), the scheduling unit **15** executes a process (steps **S120**–**S122**) for setting the queue ID of the next queue in the variable i. That is, the scheduling unit **15** compares the value of the variable i with a maximum value N of the queue ID and, if the value of the variable i is not coincident with N (step **S120**; N), adds "1" to the variable i (step **S121**). Whereas if i is coincident with N (step **S120**; Y), the scheduling unit **15** sets "1" in the variable i (step **S122**).